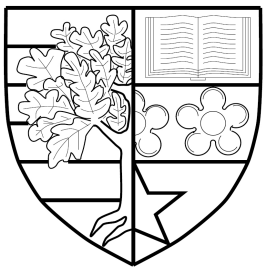
An Incremental Clustering and Associative Learning Architecture for Intelligent Robotics

Matthias Ulrich Keysermann



submitted for the degree of
Doctor of Philosophy

Heriot-Watt University
School of Mathematical and Computer Sciences

August 2015

Abstract

The ability to learn from the environment and memorise the acquired knowledge is essential for robots to become autonomous and versatile artificial companions. This thesis proposes a novel learning and memory architecture for robots, which performs associative learning and recall of sensory and actuator patterns. The approach avoids the inclusion of task-specific expert knowledge and can deal with any kind of multi-dimensional real-valued data, apart from being tolerant to noise and supporting incremental learning. The proposed architecture integrates two machine learning methods: a topology learning algorithm that performs incremental clustering, and an associative memory model that learns relationship information based on the co-occurrence of inputs.

The evaluations of both the topology learning algorithm and the associative memory model involved the memorisation of high-dimensional visual data as well as the association of symbolic data, presented simultaneously and sequentially. Moreover, the document analyses the results of two experiments in which the entire architecture was evaluated regarding its associative and incremental learning capabilities. One experiment comprised an incremental learning task with visual patterns and text labels, which was performed both in a simulated scenario and with a real robot. In a second experiment a robot learned to recognise visual patterns in the form of road signs and associated them with different configurations of its arm joints.

The thesis also discusses several learning-related aspects of the architecture and highlights strengths and weaknesses of the proposed approach. The developed architecture and corresponding findings contribute to the domains of machine learning and intelligent robotics.

Acknowledgements

Firstly, I would like to thank my supervisor, Dr. Patrícia A. Vargas, for her support and guidance throughout my Ph.D. studies. She gave me motivation and encouragement when necessary and helped me manifesting my belief in this work. Patrícia also supported my academic development by providing numerous opportunities for gathering academic experience and obtaining extra knowledge in my field of research and beyond. She was always available to discuss problems or any other issues encountered during my research.

I would also like to express my gratitude towards my second supervisor, Prof. Ruth Aylett, who guided and supported me during the initial phase of my studies. Ruth allowed me to gain valuable work experience in a European research project related to my area of research. She was available to discuss problems when needed and provided feedback which helped me to publish my work.

I acknowledge the School of Mathematical and Computer Sciences for having me awarded a Ph.D. studentship for three years. Moreover, I want to thank anyone in the department who shared their thoughts on my research and helped me to progress in my studies.

Also much appreciated is the constructive feedback I received from my examiners, Prof. Jon Timmis and Prof. David W. Corne, who helped me to improve the quality of this thesis.

I further wish to thank my partner Marta for her support, her patience, and the numerous conversations that gave me an additional opinion on my research and helped me to see things from a different perspective.

Finally, I want to thank my parents for the constant backup they provided for me during these years of studying, which made it easier to reach this point in my life.

ACADEMIC REGISTRY
Research Thesis Submission

Name:	Matthias Ulrich Keysermann		
School/PGI:	School of Mathematical and Computer Sciences (MACS)		
Version: *(i.e. First, Resubmission, Final)*	Final	Degree Sought (Award **and** Subject area)	Doctor of Philosophy (PhD) Computer Science

Declaration

In accordance with the appropriate regulations I hereby submit my thesis and I declare that:

1) the thesis embodies the results of my own work and has been composed by myself
2) where appropriate, I have made acknowledgement of the work of others and have made reference to work carried out in collaboration with other persons
3) the thesis is the correct version of the thesis for submission and is the same version as any electronic versions submitted*.
4) my thesis for the award referred to, deposited in the Heriot-Watt University Library, should be made available for loan or photocopying and be available via the Institutional Repository, subject to such conditions as the Librarian may require
5) I understand that as a student of the University I am required to abide by the Regulations of the University and to conform to its discipline.

* *Please note that it is the responsibility of the candidate to ensure that the correct version of the thesis is submitted.*

Signature of Candidate*:*		Date:	

Submission

Submitted By *(name in capitals):*	
Signature of Individual Submitting:	
Date Submitted:	

For Completion in the Student Service Centre (SSC)

Received in the SSC by *(name in capitals):*			
Method of Submission *(Handed in to SSC; posted through internal/external mail):*			
*E-thesis Submitted (**mandatory for final theses)***			
Signature:		Date:	

Please note this form should bound into the submitted thesis.

Updated February 2008, November 2008, February 2009, January 2011

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AIS Artificial Immune System.

ART Adaptive Resonance Theory.

ASH associative symmetry hypothesis.

DAIM Distributed Associative Interactive Memory.

E-SOINN Enhanced Self-Organizing Incremental Neural Network.

GAM General Associative Memory.

GH-SOM Growing Hierarchical Self-Organizing Map.

GNG Growing Neural Gas.

GNG-U Growing Neural Gas with Utility measure.

GSOM Growing Self-Organizing Map.

IAH independent association hypothesis.

ICALA Incremental Clustering & Associative Learning Architecture.

M-SOINN Modified Self-Organizing Incremental Neural Network.

MNG Merge-Neural Gas.

MSOM Merge-Self-Organizing Map.

PCA Principal Component Analysis.

RLAIS Resource Limited Artificial Immune System.

RSOM Recurrent Self-Organizing Map.

SOB "serial-order-in-a-box".

SOIAM Self-Organizing Incremental Associative Memory.

SOINN Self-Organizing Incremental Neural Network.

SOM Self-Organizing Map.

TD Temporal Difference.

TDAM Temporal Difference Associative Memory.

THSOM Temporal Hebbian Self-Organizing Map.

TNT Temporal Network for Transitions.

TODAM Theory of Distributed Associative Memory.

TOSAM Temporal-Order Sensitive Associative Memory.

List of Publications

Parts of this thesis are based on the following journal and conference papers:

- Keysermann, Matthias U. and Vargas, Patricia A. Desiderata for a Memory Model. In *Proceedings of the 12th UK Workshop on Computational Intelligence*, pages 37–44, 2012.

- Keysermann, Matthias U. and Vargas, Patricia A. An online learning system for autonomously operating robots based on spatial and temporal associative learning. In *International IEEE/EPSRC Workshop on Autonomous Cognitive Robotics*, University of Stirling, Stirling, UK, 2014.

- Keysermann, Matthias U. and Vargas, Patrícia A. A learning and memory architecture for robot companions based on incremental associative learning. In *Proceedings of the IEEE RO-MAN '14 Workshop on Developmental and bio-inspired approaches for memory and emotion modelling in cognitive robotics*, pages 13–14, 2014.

- Keysermann, Matthias U. ICALA: Incremental Clustering and Associative Learning Architecture. In *Adaptive and Intelligent Systems*, volume 8779 of *Lecture Notes in Computer Science*, pages 70–79, 2014. doi:10.1007/978-3-319-11298-5_8.

- Keysermann, Matthias U. and Vargas, Patrícia A. Towards Autonomous Robots Via an Incremental Clustering and Associative Learning Architecture. *Cognitive Computation*, 7(4):414–433, 2015. doi:10.1007/s12559-014-9311-y.

Chapter 1

Introduction

A robot can be defined as a machine that is equipped with sensors and actuators and is able to sense, think and act (Bekey, 2005). The field of robotics comprises many different types, ranging from industrial robots in manufacturing to mobile robots that operate in complex, dynamic and sometimes unstructured environments. Autonomy and robustness are required in particular by robots that can accompany and assist humans in a variety of tasks (Vargas et al., 2014). These types of robots have been recently called artificial companions (Dautenhahn et al., 2005), and as such, they are expected to be able to live and/or interact with humans for an extended period of time (Castellano et al., 2008; Aylett et al., 2011).

In the light of keeping robots as artificial companions, special attention needs to be given to the underlying architecture that equips these companions with the capabilities to learn and execute different behaviours. Therefore, a specialised architecture built to fulfil only certain tasks is not adequate. The development and expression of intelligent behaviour in a wide variety of different situations can only be achieved with a general and versatile architecture. Such an architecture cannot draw upon a static pre-specified amount of knowledge, which would restrict its applicability to only particular domains. Even if a learning architecture is trainable but does not update its knowledge during runtime, again its applicability is limited to the data used for training (Thórisson, 2012). The stated claims demand for an adaptive architecture with an expandable memory that can incrementally store information and comprehensively build up experience.

One way of accomplishing this is to empower the robot with an architecture which aims for Artificial General Intelligence (Thórisson, 2012). In this way, a robot will not be restricted to operate only within a particular domain and also to execute only a limited range of tasks. For a robot to cope with new, unseen scenarios, the structures used for knowledge representation must be general enough to deal with all sorts of data. Similarly, this holds for the chosen learning and reasoning methods. A high level of generality can be achieved by starting from non-specific low-level paradigms, like associative learning or Hebbian learning (Hebb, 1949; O'Reilly and

Munakata, 2000), to more complex yet still generic processes.

1.1 Cognitive Architectures

General frameworks for cognition are given by cognitive architectures (Sun, 2004; Taatgen and Anderson, 2009; Langley et al., 2009). Langley et al. (2009) state that a cognitive architecture should be capable of recognising and categorising information perceived from the environment, assessing a situation correctly, making decisions and reasonably choosing from alternative actions, executing the chosen ones, predicting and monitoring future outcomes. Therefore, a cognitive architecture should be able to solve problems, in particular to use planning, and reason about beliefs or assumptions made, which can be based on remembered past situations and the reflection made upon. Furthermore, a robot driven by such an architecture might be required to interact and communicate with other robots in order to transfer and share knowledge. According to Langley et al. (2009), the main questions to be answered when designing a cognitive architecture are: How can knowledge be acquired? How should it be represented within the architecture? How is it organised and refined? Even if left unanswered by the authors, this thesis agrees about the significance of these questions and presents a practical solution.

Taatgen and Anderson (2009) advocate that a cognitive architecture should be as simple as possible. Having a high number of predefined parameters does not only reduce the direct applicability and usability of the whole architecture but also allows the underlying model to perform in a way such that the different results produced can fit various experimental data (Taatgen and Anderson, 2009). This makes it hard to measure the overall plausibility of the model, as the architecture can be manipulated by a setup that is tailored to the current task. The resulting behaviour would not be achieved in an entirely autonomous way by the architecture itself. Apart from minimising the number of parameters, it is also important to minimise the task-specific knowledge given in advance or built into the architecture. Only giving up scope on a particular task or domain allows the creation of universal, general models, which can adapt and cope with a large variety of different situations (Taatgen and Anderson, 2009).

A shortcoming of existing architectures like ACT-R (Anderson and Lebiere, 1998; Anderson et al., 2004) or SOAR (Newell, 1990; Laird, 2012) is that they require the user to specify knowledge in advance, for instance, goals and possible actions in a discretized form. Especially when dealing with sensory data, these architectures are cumbersome to set up and not easily transferable to various different domains. As claimed by Sun (2004), instead of focusing on laboratory or toy problem tasks, cognitive architectures should focus on tasks and activities encountered in everyday life (ecological realism). An architecture should also account for a wide range

of sophistication levels as found in different living beings – making it possible to continuously model cognitive processes from animal level to human level (bio-evolutionary realism). This includes the avoidance of highly sophisticated mechanisms which are not reflected in animal behaviour. However, essential characteristics of human behaviour should still be captured (cognitive realism).

Different cognitive assumptions can bias an architecture and its behaviour towards mimicking certain characteristics or others. It is also necessary to find a basic subset of cognitive assumptions to serve for the development of future architectures, which enables comparison amongst these architectures. No consensus has been found yet over the various memory systems within a cognitive architecture suggested throughout existing literature (Ratcliff and McKoon, 2000; Sun, 2004; Ho et al., 2008). It is unclear which dichotomies are essential, For instance, should a cognitive architecture distinguish between implicit and explicit learning? Are procedural and declarative knowledge to be treated differently? Moreover, other issues in the domain of cognitive modelling still remain open for further research, such as the integration of emotions, communication among agents, the categorisation and understanding as well as the encoding of information, and how different formalisms can be linked to each other.

In order to allow an architecture to successfully cope with unfamiliar environments, robust and flexible learning mechanisms are needed (Langley et al., 2009). Still also under investigation is which impact the physical embodiment has on cognitive processes (Pfeifer and Scheier, 1999; Pfeifer and Bongard, 2006). For a precise and comprehensive evaluation of cognitive architectures, complex and realistic environments need to be created and experiments for systematic testing have to be designed carefully (Langley et al., 2009).

1.2 Desiderata for a Learning and Memory Architecture

While a cognitive architecture comprises many different aspects of cognition, the parts concerning learning and memory are usually the core components and thus play a central role. This thesis focuses on these key aspects and, influenced by the mentioned demands on cognitive architectures, starts by formulating desirable characteristics in regard to learning and memory. In the following, general desiderata for a learning and memory architecture are given, as described by Keysermann and Vargas (2012).

- **Universality:** The architecture should be able to handle any kind of information while a constant stream of inputs can be fed into it. The architecture should not require the inputs to be preprocessed or restructured in an informed way. The informed restructuring of inputs would mean to utilise external

knowledge about the data, which has neither been acquired nor learned by the architecture. Moreover, no prior information about specific domains should be built into the architecture. Instead, it should be possible to use the architecture in a variety of different environments and on many different tasks.

- **Adaptability:** The architecture should be able to adapt to changed circumstances and should be able to alter or modify previously learned knowledge as the environment changes. In particular, stored concepts or created associations may become outdated and inaccurate after some time. In addition, and in line with the claim for universality, the architecture should be able to deal with incoming information of a different nature than the information processed so far. The long-term behaviour generated by the architecture should be insensitive to minor changes of the environment, such that radical changes and heavy behavioural oscillations are avoided. A gradual modification of long-term objectives or overall goals should be allowed to occur as context and external stimuli change.

- **Organisation:** The knowledge stored by the architecture should be automatically organised such that this organisation supports the selective retrieval of information and helps to maintain the architecture's performance. The information organisation process can be of the form of linking related and relevant information and, optionally, arrange it in a hierarchical order. This can involve the abstraction or generalisation of information. To meet the claim of universality, only the information learned by the architecture itself should be utilised. The stored information should be content-addressable, which means that neither indices nor any other auxiliary mappings or metadata can be used for accessing it. Abstraction or generalisation may enhance the selective retrieval of information.

- **Scalability:** The architecture should be able to cope with an increasing amount of data while the computational effort for storing, retrieving and reorganising the information should not become significantly higher as more data is included. Thus, especially the mechanisms for the storage and retrieval of information need to be efficient. High performance may be assured by some sort of abstraction or generalisation mechanism, which helps to reduce the amount of data that needs to be processed. A scalable architecture should be able to deal with more and more knowledge being added to its memory. If an architecture results in the driven system to become slower when the amount of stored knowledge increases, it suffers from the so-called *utility problem* (Minton, 1990), which is still an open issue for research (Langley et al., 2009).

Ideally, a learning and memory architecture should meet all of the formulated desiderata. But most of the aforementioned criteria pose difficult research challenges

on their own or would require an extensive amount of time to study. For instance, principles of universality can be built into the data processing methods, but the evaluation of a sufficient level of generality can only be proven over time in many different experiments. Also the possibility of an ever-increasing knowledge store usually comes at the cost of computational effort to process the data, as the underlying hardware is still limited. Hence, not each single property can be taken into consideration and it is necessary to focus on the most essential points.

1.3 Aims and Objectives

Hypothesis:

A learning and memory architecture that uses only unsupervised learning methods without any domain knowledge allows a robot to learn reactive behaviours.

The overall aim of this thesis is to create a universally-applicable, adaptive learning and memory architecture for robots, which automatically structures the inputs as they are perceived from the robot's sensors.

Such an architecture will not require a pre-specified fixed training dataset that needs to be learned completely before the knowledge can be applied. Instead, inputs are constantly perceived, processed, stored, categorised or otherwise altered and incrementally form a knowledge base, which can be permanently accessed to react and control the exhibited behaviour of the robot.

In order to allow inputs of any sort, a fixed and predefined processing pipeline for the input data is not sufficient. Thus, all processing steps are general enough to deal with any kind of input data. This requirement also ensures that the algorithms are not restricted to only a particular modality or sensor type. A top-down approach can easily bias the learning procedure towards a specific task or dataset. Thus, a bottom-up approach is chosen to formulate basic general principles that can be tested in simple learning situations.

With the hard constraint of being universal, the amount of domain knowledge built into the architecture is kept to a minimum. Otherwise, the developer would equip the architecture with knowledge being to their observations. This knowledge would have been extracted and processed by the developer, who is already capable of intelligently transforming the incoming information into meaningful knowledge. The architecture would not develop this knowledge on its own and would possibly be not even capable of doing so. Thus, a major part of the learning process would not be performed by the architecture itself.

Hard-coded knowledge does not reflect the intelligence of the system driven by such an architecture but only the expertise of its creator. A learning architecture for robots must be able to learn only by the robot's sensory perceptions but no other information sources. Supervised learning algorithms cannot be used as they would

require additional information about the input data, such as class labels (Witten et al., 2011). Instead, the learning algorithms should work fully unsupervised.

Aiming for a high usability, the architecture is easily applicable without requiring the user to have a deeper knowledge about the internal functionality or any other specific details. Additionally, modularity in the design of the architecture allows to include any combinations of input and output modules in the form of different sensors and actuators. In this way, the architecture can be compatible with multiple robotic platforms.

While some of these aims set demands on the desired architecture, others pose constraints. Before being able to accept or reject the stated hypothesis, different aspects must be addressed such as the construction of the architecture, its application to robot learning and the resulting behaviour. These are separate research questions that need to be answered. The main research questions this thesis addresses are:

1. How can a versatile learning and memory architecture be constructed without requiring domain-specific processing or relying on domain-specific information?

2. How can such an architecture be applied to robot learning while the robot's sensors are the only source of information?

3. Which type of behaviours can a robot learn while solely relying on generic unsupervised learning methods?

1.4 Contribution of this Thesis

The development of a learning and memory architecture for robots contributes mainly to the domains of machine learning and robotics. This thesis formulates adequate learning algorithms and specifies necessary structures to hold the learned information, which allows to investigate low level requirements for learning and for the emergence of behaviours. The research results provide a useful input to the robotics community as the developed components can be applied to various robot tasks.

The contributed learning and memory architecture has the following features:

- The proposed architecture supports incremental learning and uses dynamically growing structures for storing information.

  In incremental learning scenarios the total extent of the incoming data cannot always be determined beforehand. Hence, the representational capacities of the included learning methods should not be limited in terms of the amount of information that can be represented. Also additional information like the number of classes or categories that are included in the input data must be assumed as unknown. Although several popular algorithms for clustering require to specify the desired number of clusters in advance (Bishop, 2007; Witten

et al., 2011), this should not be a requirement when the clustering is performed incrementally (Keysermann, 2014). Similarly, many neural network approaches work with a fixed number of nodes (Haykin, 2008) and, as a consequence, suffer from the phenomenon of *catastrophic forgetting* (Section 2.1.2). This problem can be avoided by utilising a dynamically growing network structure (Keysermann and Vargas, 2014b; Keysermann, 2014).

- The proposed architecture tolerates noise and allows a flexible timing of the perceived inputs.

  Especially important in regard to robot learning, where inputs originate from hardware sensors, is the capability of an algorithm to deal with noise in the inputs. Such an algorithm should reliably recognise previously encountered inputs even if the perceived data pattern is not identical to the one that has been learned initially. Moreover, if inputs are generated by an external source, the precise timing of inputs can vary. An adequate learning method should account for this by tolerating short gaps between the perceived inputs (Keysermann and Vargas, 2014a, 2015).

- The proposed architecture avoids the requirement of technical knowledge and can learn from the received sensory inputs alone.

  A human user should be able to teach a robot new behaviours without having much technical knowledge about the underlying architecture itself. It is undesirable to re-engineer parts of the architecture or to translate new data into a system-specific format. More generally, users should have the possibility of teaching a robot novel information without having expert knowledge about the incorporated learning algorithms or data structures (Nicolescu and Matarić, 2003). Robot learning should happen only by interaction and the corresponding learning and memory architecture should work solely on the inputs received from the robot's sensors (Keysermann and Vargas, 2014a, 2015).

Corresponding details of the previously stated claims will be elaborated throughout the thesis.

1.5 Outline of the Thesis

Chapter 2 reviews the relevant literature and related research. The proposed architecture and corresponding algorithms are described in Chapter 3. In Chapters 4 and 5, the two main components of the architecture are separately evaluated. Chapter 6 provides a validation of the entire architecture, which was conducted in two different experiments involving a real robot. Then the overall results are discussed in a more general context in Chapter 7. Finally, Chapter 8 concludes the thesis and gives an outlook onto future work.

Chapter 2

Learning and Memory Models for Intelligent Robots

This chapter examines robot learning and suitable memory mechanisms for intelligent robots. Several machine learning algorithms are described to provide a basis for the approach taken in this thesis. Also some background from research in psychology is given to support certain modelling aspects.

In the following, first important aspects of learning and memory for robots are described (Section 2.1). Thereafter, Section 2.2 presents existing methods for online incremental learning and Section 2.3 discusses associative learning as a form of unsupervised learning. Section 2.3.3 gives an overview of approaches to model serial order recall and provides different perspectives on the emergence of asymmetric recall effects.

2.1 Learning and Memory in Robots

Two key aspects of intelligent robotics are learning (Arkin, 1998; Murphy, 2000; Bekey, 2005) and memory (Pfeifer and Scheier, 1999). Both of these concepts are heavily intertwined and together they fundamentally support the development of the sophisticated behaviours an intelligent robot could produce. An embodied robot can perceive and interact with its environment via its given sensors and actuators. In this regard, robot learning can be seen as the process of utilising perceptions and interactions with the environment to derive information about the robot's surroundings and to improve its behaviours regarding a certain task. Memory is essential for gathering experience and storing the acquired knowledge (Baddeley, 1997; Anderson, 1999). The use of learning and memory mechanisms allows a robot to adapt to the infinite range of circumstances it can encounter in the real world, and produce the appropriate behaviours for any given situation. Thus, an architecture for a robot system that aims for autonomy and a high level of interaction complexity

needs to learn from its inputs and maintain the derived knowledge over an extended period of time.

Bekey (2005) sees robot learning as a basis for an improvement in task performance or, similarly, the decrease in error rate with increasing experience. Also Arkin (1998) defines learning in terms of the functional implications it has on the performance of a robot or agent within its environment, namely increasing the robot's effectiveness. Pfeifer and Scheier (1999) argue that, in the domain of robotics, a learning algorithm requires specific essential properties. As an intelligent robot operates in a real environment, the learned knowledge must be grounded. Linking internally created representations to real world entities enables a robot to actively interact with its surroundings and effectively manipulate objects in its environment. When working with the actual data of sensors and actuators, a corresponding learning algorithm should be tolerant to the noise present in these inputs and be able to cope with minor perceptional variations. Additionally, the knowledge must be extracted fast and, thus, be ready for the evaluation of further actions. Many situations demand an immediate response by the robot and, hence, the algorithm should work in real-time and allow the robot to respond within hundreds of milliseconds. In many situations soft real-time is adequate, i.e. exceeding the demanded response time does not have negative consequences. However, some situations may demand hard real-time as the robot could be damaged otherwise. In this regard, learning should happen online and simultaneously to any other processing tasks the robot is executing. An intelligent robot cannot afford to miss significant experience while performing a specific action. Similarly, a learning algorithm must be able to extend its knowledge database as new perceptions are encountered via the robot's sensors. A learning algorithm, which works incrementally, can incorporate new instances without losing already stored knowledge and can enable a robot to operate in a dynamic environment. Also Mahadevan (1996) emphasises the importance of incremental and online learning for robots. Moreover, the robot must be able to respond in real-time, apart from being able to deal with the noise created by its sensors.

The memory system of an autonomous robot is likely to receive a lot of sensory data about the surrounding world as input. Even if a form of preprocessing is applied to the input data, for instance discretization or categorisation, there can still be a lot of redundant and useless information. Building memory processes such as generalisation and forgetting into cognitive architectures for robots keeps the total amount of stored information at a moderate level and facilitates fast processing and rapid access. Furthermore, interaction with a robot with these capabilities is perceived as more natural and makes the robot appear as a companion rather than a pure information organiser (Lim et al., 2011). Apart from generalisation and forgetting, another highly important memory process is association (Eysenck and Keane, 2005; Parkin, 2006), which allows to model relationships between memories.

These memories can be represented in network structures, to which graph algorithms and spreading functions can be applied. The process of generalisation captures common concepts of several single memories and abstracts from specific details. Computationally, this process can include dimensionality reduction, some sort of clustering, building a representative prototype and also pruning or merging of the stored data structures. Forgetting considers theories like trace decay, repression and interference (Vargas et al., 2009; Ho et al., 2009; Vargas et al., 2011), and deals with the consolidation of frequently recalled memories (Gais and Born, 2004). Functions for forgetting over time, tagging of memories and methods for restructuring the stored data can be used to model this aspect. The proposed approach (Chapter 3) incorporates methods for the generalisation of perceptions and the association of formed categories. Although the developed memory model contains characteristics of trace decay forgetting, theories of forgetting are not addressed in detail.

2.1.1 Symbol Grounding

The input patterns as captured by a robot's sensors are the first data that could be stored in the robot's memory. This data is suitable to be used with pattern recognition methods. However, for modelling higher cognitive processes, approaches that operate on discrete symbols seem to be more appropriate, as pointed out by Haikonen (2009). The author argues that, in order to bridge the gap between tasks like pattern recognition or classification and processes attributed to higher cognition, sub-symbolically operating neural networks must be combined with symbolic approaches. Such a hybrid approach could be a connectionist architecture that can also process symbolic information. In an approach like this, sensory patterns are associated with their corresponding symbols (Haikonen, 2009). If such patterns originate from perceptions, then the associated symbols are linked to their real-world representations. The architecture should be the owner of these formed meanings (Haikonen, 2009). Thus, the architecture must form these concepts on its own, based on the patterns it receives as input. Similarly for sub-symbolic representations, Wichert (2009) argues that there must be a relationship between a real-world object or state and its representation, established through sensors or senses. In order to combine both neural and symbolic information processing, Velik (2007) and Velik and Bruckner (2008) suggest a multimodal approach where perceptual information is hierarchically transformed into symbolic concepts. In this approach the data originates from multiple receptors and undergoes various pre-specified processing steps before the information of different modalities is combined into multimodal concepts. Instead of using modality-specific steps for transforming sensory data into symbolic data, the architecture described in Chapter 3 employs a topology learning algorithm as a more general method of abstracting the input data, such that it becomes directly usable with associative learning methods.

The principle of connecting entities of the environment to internally formed symbols is known as symbol grounding (Taddeo and Floridi, 2005). Originally defined by Harnad (1990), the *symbol grounding problem* seeks to find a way for giving meaning to the created symbols. As symbols alone do not lead to meaning, they must be grounded in something with meaning. In the case of embodied autonomous agents, the grounding can happen by relating actual sensor readings to internal symbols. These symbols are then meaningful for an agent as they represent the actual environment the agent interacts with, which, in turn, has influence on the agent itself. Harnad (1990) suggests to use a hybrid bottom-up approach, consisting of a connectionist model combined with a symbolic system. In this approach, the many perceptions of real-world objects lead to the creation of iconic representations. By extracting the invariant features of each of these objects, categorical representations can be formed. Finally, symbolic representations, which are grounded in these former non-symbolic representations, can be linked or connected, which allows to describe membership or relationship information, and can be combined and used by a symbol manipulation system.

If a cognitive architecture seeks to solve the symbol grounding problem, then the grounding process must be done exclusively and autonomously by the architecture itself (Taddeo and Floridi, 2005). In this regard, no hard-coded knowledge or any other semantic resources can be included in the architecture (no innatism). Similarly, no semantic knowledge of any form can later be injected into the architecture from an external source, such as a human who already acquired this knowledge before (no externalism). The architecture must learn to ground its symbols entirely on its own by using the available sensing and acting capabilities. Taddeo and Floridi (2005) call these formulations the *zero commitment condition*. Apart from fulfilling this condition, a symbol grounding system should also have the following features. It should include both a bottom-up sensorimotor approach and a top-down feedback approach, have representational capacities, categorical or abstracting capacities, as well as communication capacities and follow an evolutionary approach (Taddeo and Floridi, 2005).

2.1.2 Incremental Learning

For effectively working with unstructured noisy data, for instance data as perceived by sensors, a categorisation of the data can be highly beneficial. Similarly, the process of generalisation allows to abstract from small variations in the perceived inputs while focusing on the most significant aspects. If the nature of the data is unknown in advance, it is not possible to apply any data-specific preprocessing mechanism or any algorithm that is tailored to a particular domain. Thus, only the received input data on their own can be used to drive generalisation and categorisation. These processes cannot rely on strong assumptions about the structure or format of the data, except

of possibly a fixed dimensionality and the perception of inputs happening in regular intervals. Still, the architecture must be able to separate and group the data by forming clusters, which then allows to extract abstract categories.

While the nature of the incoming data can be unknown in advance, also unclear can be the total extent of the data. In other words, the amount of data is not necessarily limited or of a fixed size. In particular, if entirely new inputs can be received at any time, the chosen representation for storing and clustering the data must be of dynamic size. Allowing for an infinitely large training set, a suitable clustering algorithm needs to be incremental and work online, i.e. update its representations whenever data is received. If the distribution of inputs can change over time (e.g. in dynamic environments), such an algorithm must be adaptive. The topology learning algorithm used in the proposed architecture fulfils the mentioned requirements (Section 3.1). The topology can grow and shrink dynamically while the algorithm learns new data incrementally.

For a continuously learning memory model, which is to be used in a variety of different environments and situations, the entire amount of data to be learned cannot be determined in advance. In other words, the respective training dataset can grow incrementally and is potentially unlimited in its size. A learning architecture appropriate for the given circumstances needs to be capable of dealing with such a dataset. In particular, a model of fixed storage capacity is unsuitable. The problem of models with a fixed structure for information storage, e.g. a neural network with a fixed number of neurons, is their limited topology – independently of the dimension of the model. In order to cope with the demands of an ever-growing knowledge store, the proposed associative memory model (Section 3.2) utilises an incremental structure for the employed network. On the other hand, adopting an expandable and growing structure can lead to runtime problems under heavy load – due to the given limitations of the underlying hardware. The proposed model handles this issue by applying a clean-up procedure. This mechanism avoids the formation of infinitely large networks by pruning insignificant knowledge.

Yet, the model should be able to adapt to different environments and contexts without unlearning important information. This problem is known as the *stability-plasticity dilemma* (Grossberg, 1987, 2013) and gives another reason for using a dynamic network size over a fixed-size storage. Models with a limited storage capacity can suffer from *catastrophic forgetting* (or *catastrophic interference*) (Sharkey and Sharkey, 1995; French, 1999). Already learned knowledge can be overwritten by incoming information. This effect is especially facilitated with extensively distributed methods of storing information.

2.2 Methods for Online Incremental Learning

Working with robots adds special requirements to the used learning methods, for these methods should work online and incrementally (Pfeifer and Scheier, 1999). A system with the ability to learn online can directly incorporate newly acquired information and include it in further decision making. Instead of requiring an extended batch of input data, an online learning system updates its knowledge database after every single instance. Thus, it can learn smaller amounts of data and directly recall the stored knowledge during the ongoing learning process. This aspect becomes crucial if the entire training dataset cannot be determined in advance, e.g. if certain information only becomes available in the future or if the system must deal with a continuous stream of new inputs. When learning incrementally, new inputs extend the already stored knowledge without having to sacrifice information that has been learned previously. Incremental methods are also more efficient in terms of learning time because only additional instances need to be incorporated to update the stored knowledge, instead of relearning the entire training dataset. The following sections describe a number of incremental learning algorithms, most of which also work online. Although originally intended as topology learning algorithms, they are excellent candidates for incremental clustering (Du, 2010; Bouchachia, 2011).

2.2.1 Growing Self-Organizing Maps

The Self-Organizing Map (SOM) (Kohonen, 2001) is a method for learning topologies of the input data. As SOM uses a fixed-size network, it cannot always represent the structure of input data appropriately and often it is hard to determine the optimal size of the topology in advance (Alahakoon et al., 2000; Dittenbach et al., 2000, 2002; Zhou and Fu, 2005). To eliminate these shortcomings, variants of SOM have been proposed were the topology can grow dynamically.

Bauer and Villmann (1997) describe an approach for a growing SOM where the topology is stored as a generalised hypercube. Initially, the algorithm starts with a configuration with two nodes (in a one-dimensional topology). The topology can grow by adding nodes along existing dimensions or by adding a new dimension to the hypercube. In order to decide in which direction to grow, the reconstruction error is decomposed along the different dimensions and nodes are added in the direction with the largest error amplitudes on average. Whenever the topological structure has changed (i.e. after each growth step), relearning is required to adjust the topological map. The algorithm has been evaluated regarding its neighbourhood preserveration properties on image data and speech data (Bauer and Villmann, 1997) as well as on time series data, remote sensoring data and medical data (Villmann and Bauer, 1998).

Alahakoon et al. (1998) introduced the Growing Self-Organizing Map (GSOM) which uses a two-dimensional (sparse) topology that can be dynamically expanded by adding nodes. The topological growth process in GSOM depends on the accumulated error of each node and a predefined threshold value. If the error of a node exceeds this threshold, GSOM inserts new nodes in case the corresponding node is a boundary node. For non-boundary nodes, the weight vectors of the corresponding node and its neighbours are redistributed (Alahakoon et al., 1998, 2000). In order to control the granularity of the resulting topology, the growth threshold can be set based on a spread factor and independent of the dimensionality of the input data. This allows to train multiple topological maps with different spread factors and organize them in a hierarchical order. Topologies with high spread factors provide a coarse clustering, whereas low spread factors lead to a fine-grained clustering result (Alahakoon et al., 2000; De Silva et al., 2007).

Dittenbach et al. (2000) proposed the Growing Hierarchical Self-Organizing Map (GH-SOM), which uses multiple layers of two-dimensional SOMs where each SOM can incrementally grow on its own. This topological growth process happens after a fixed number of training iterations. GH-SOM determines the node with the highest accumulated error and its most dissimilar neighbour node. Then GH-SOM extends the topology by inserting either a new row or a new column of nodes in between the determined nodes. Once a certain level of granularity has been reached, GH-SOM checks if hierarchical growth is required. If the quantization error of a node exceeds a certain threshold value, then GH-SOM extends the hierarchical structure by adding a new SOM in deeper layer. The new SOM is trained until its mean quantization error is reduced to a fraction of its parent node. GH-SOM is especially useful for hierarchical clustering tasks and was successfully applied to document clustering (Dittenbach et al., 2000, 2002; Rauber et al., 2002).

A GSOM variant described by Zhou and Fu (2005) uses a two-dimensional SOM where nodes are connected in a triangular way. For each input the algorithm adds the distance to the weight vector of the respective winning node to its accumulated error. For growing the topology, the node with the highest accumulated error is determined as well as its most-distant neighbour. Then a new node is inserted halfway between these nodes and the local connectivity is adjusted accordingly. In order to detect clusters, the algorithm periodically removes nodes in regions of low density based on a threshold parameter (Zhou and Fu, 2005).

2.2.2 Growing Neural Gas

The Growing Neural Gas (GNG) was introduced by Fritzke (1995) for both supervised and unsupervised learning. Based on an approach proposed by Martinetz and Schulten (1994), the algorithm uses competitive Hebbian learning to create a network structure that represents the topology of the inputs received. Inputs are represented by nodes

and these nodes' positions are adjusted while input data is received. Edges connect the created nodes incrementally to form a structure. This structure arises from the distribution of the data in the input space. GNG does not require any parameters that decrease over time. Hence, it is particularly suitable for problems where the size of the training dataset is not known in advance. In these cases it cannot be clearly determined when to stop learning and when the algorithm should converge.

The algorithm works as follows. The network is initialised with two nodes randomly placed in the input space. Whenever GNG receives an input, the two nodes in the network that are closest to the input are determined. The Euclidean distance from the nearest node to the input pattern is computed and this local error is added to the accumulative error for this particular node. The position of this node is adjusted by a fraction to move closer to the actual input, and also connected nodes are moved towards the input. In addition, for each of the involved edges an *age* parameter is incremented which counts how often connected nodes were moved. The age of the edge between the nearest node and the second-nearest node is reset to 0. In case these nodes are not connected, the corresponding edge is created. In order to clean up the network structure, edges with an age greater than a predefined threshold as well as isolated nodes are removed. Furthermore, the error of all remaining nodes is weakened by multiplicatively decreasing it. After a predefined number of inputs, the algorithm tries to better fit the input data in areas with high error. This is done by determining the network node with the highest error as well as the maximum-error node amongst the nodes connected to the former. Then a new node is created between these two high-error nodes, is connected to both of them and the original edge between the two high-error nodes is removed. Figure 2.1 shows an example topology created by GNG.

In order to cope with non-stationary data distributions, Fritzke (1997) extended GNG with a utility measure, resulting in the Growing Neural Gas with Utility measure (GNG-U). This extension allows the algorithm to remove nodes in areas where no inputs have been received for a long time and enables the learned topology to correctly adapt to new inputs. If the distribution of the data changes over time, GNG-U removes nodes that are no longer representative enough such that the topology contains only currently relevant inputs.

GNG-U works like GNG but additionally stores a local utility value for each node. Whenever an input is received, the utility value of the node nearest to the input is increased. When removing nodes, all utility values are taken into account and the node with the lowest utility is evaluated for removal. The removal criterion is based both on the utility value of the node to be removed and the maximum error amongst all nodes in the network. Similarly to the process of decreasing the error values, also all utility values decay while inputs are received.

Several input processing principles of GNG are also included in a later algorithm

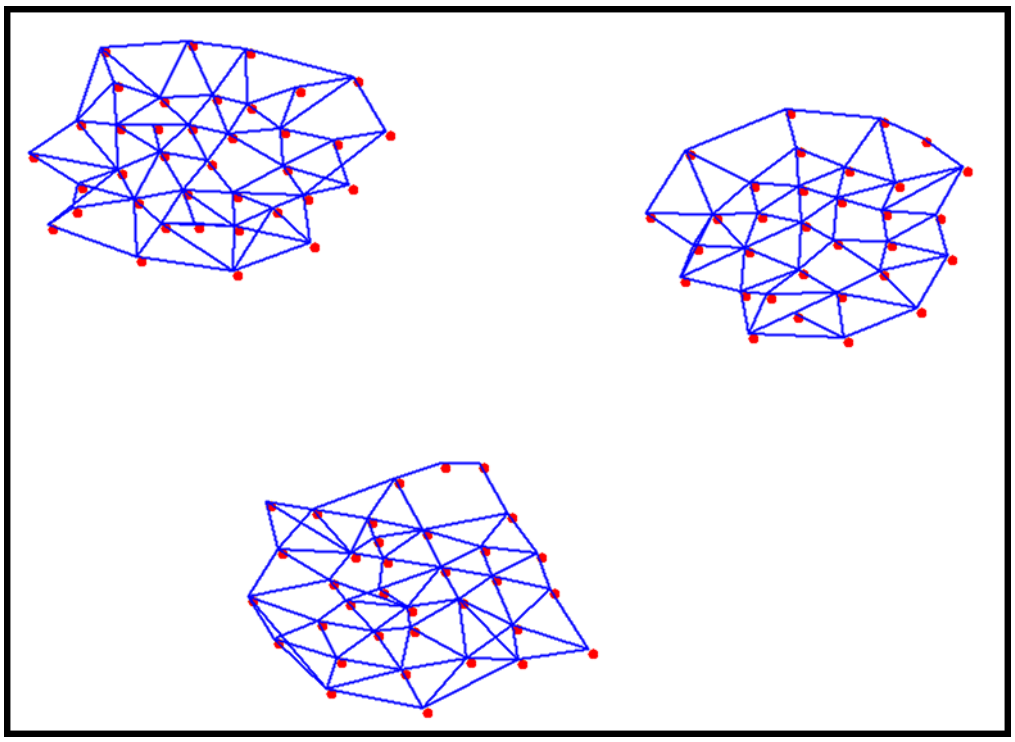


Figure 2.1: Example of a topological structure learned by the GNG algorithm whereby the data samples originated from three different points in a two-dimensional space with Gaussian-distributed noise added to the original inputs. The underlying data distribution can be clearly seen in the emerged topological structure with three separate clusters, each represented by a lattice of connected nodes.

called Self-Organizing Incremental Neural Network, which is described in the following section.

2.2.3 Self-Organizing Incremental Neural Network

The Self-Organizing Incremental Neural Network (SOINN) is a topology learning algorithm introduced by Furao and Hasegawa (2006), which modified GNG. When an input is received by SOINN, the insertion of a new node depends on the local topological structure of the region into which the new node is to be placed. In this way, SOINN addresses the permanent increase in the number of nodes occurring with GNG (Furao and Hasegawa, 2006). Accordingly, SOINN only inserts new nodes in regions with high error if this process results in a lower error. Furthermore, SOINN employs a threshold-based removal process for nodes with only one neighbour, in addition to the removal of isolated nodes. SOINN uses two layers. Similarly to GNG, the first layer builds a topological structure of the data while inputs are received. In order to obtain a better topology, SOINN applies a second training phase that builds a refined structure in another layer. Furao and Hasegawa (2006) tested SOINN on clustering tasks for both low- and high-dimensional data as well as on vector quantization and achieved better results compared to GNG. Yet SOINN suffers from some drawbacks. Apart from having problems of separating overlapping clusters in

high-density regions, the algorithm relies on three manually-set parameters whose optimal settings vary from task to task.

Learning in SOINN works as follows. The algorithm initialises the topology to contain two nodes which are randomly positioned in the input space. When SOINN receives an input, the algorithm determines the two nodes nearest to that input. For these two nodes *similarity thresholds* are calculated based on the maximum distance to their respective neighbours (directly connected nodes); or the minimum distance to other nodes in case the respective node has no neighbours. Furthermore, the Euclidean distances from the input to each of these nodes are calculated. If any of these distances exceeds the respective similarity threshold, SOINN creates a new node at the position of the input and proceeds with the next input. Otherwise, the algorithm creates an edge between the two nearest nodes (if such an edge did not exist before) and the corresponding *age* parameter is reset to 0. Then the age values of all edges emanating from the nearest node are incremented. For this nearest node also the *local accumulated error* is increased by the node's Euclidean distance to the current input. Additionally, a counter is incremented which stores the number of times this node has been nearest node, referred to as the *number of signals*. Afterwards, SOINN adjusts the positions of both the node itself and its neighbours to be closer to the current input. Finally, the algorithm removes edges whose age is greater than a pre-specified threshold. Regularly, after a certain number of inputs, the algorithm tries to improve the topology by inserting a new node next to the node with the highest error and by removing nodes with no more than one neighbour. The insertion of a new node depends on both the error and the number of signals of the maximum-error node, its maximum-error neighbour as well as the inherited values of the new node; SOINN only inserts a new node if the error per signals ratio can be decreased for all involved nodes. Moreover, the algorithm removes weakly connected nodes. While isolated nodes are always removed in this step, the removal of nodes with only one neighbour depends on a threshold criterion based on the number of signals of the node in question and the average number of signals over all nodes in the network. When training the second layer, SOINN requires topological information of the first layer and, in particular, the (preliminary) cluster structure. As this process requires the first layer to be trained (at least to some extent), it can be executed only in regular intervals and first layer learning is interrupted. Such an interleaved learning process does not allow for permanently ongoing online learning. An example of a SOINN first-layer topology is given in Figure 2.2.

Furao et al. (2007) published an improved variant of SOINN, the Enhanced Self-Organizing Incremental Neural Network (E-SOINN), which tries to overcome the limitations of SOINN. E-SOINN requires fewer parameters than SOINN and uses only one layer to store the final topological structure of the inputs. Compared to SOINN, E-SOINN is more stable with respect to the number of clusters it creates

for a given dataset. In particular, E-SOINN shows better clustering results when the inputs originate from a non-stationary data distribution. As E-SOINN uses only one layer, no interruption for learning another layer is required and, thus, it is suitable for online learning.

The training algorithm of E-SOINN works similarly to SOINN. But instead of using a trained topology which is then used to train a second layer, in E-SOINN each node is assigned to a so-called *subclass*. This process happens while the training of the first (and only) layer is ongoing and serves as a preliminary cluster assignment. Different subclasses are determined based on local density distributions. The density at the position of a node is represented by the number of times the node has been nearest to an input. Each node with local maximum density is assigned to a different subclass and all other local nodes are given the same assignment. Edges between nodes with different subclass labels can then be removed in order to split up a connected node structure which actually emerged from two different input classes. Subclass labels are also taken into account for deciding whether to create an edge between the two nodes nearest to an input. Furao et al. (2007) remain unclear about how to exactly implement the procedure of subclass assignment.

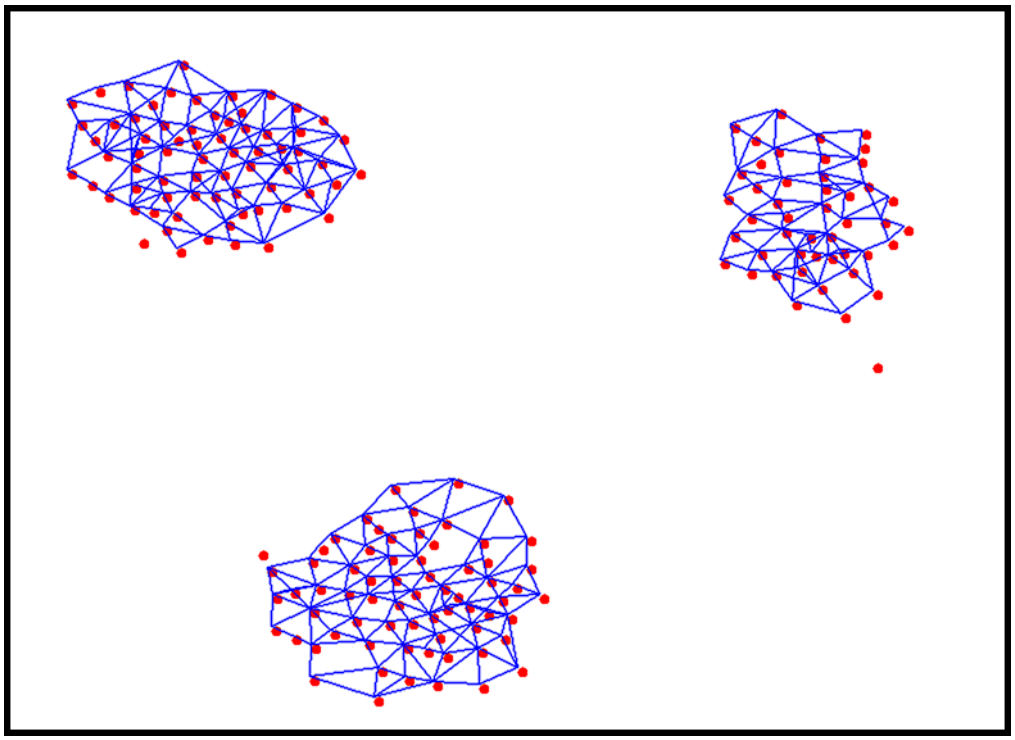


Figure 2.2: Example of a topological structure learned by the SOINN algorithm whereby the data samples originated from three different points in a two-dimensional space with Gaussian-distributed noise added to the original inputs. The underlying data distribution can be clearly seen in the emerged topological structure with three separate clusters, each represented by a lattice of connected nodes. Any isolated nodes are removed by the algorithm in regular intervals.

In order to link pairs of patterns, Sudo et al. (2009) used a modified SOINN algorithm as an associative memory, the Self-Organizing Incremental Associative

Memory (SOIAM). Two values are associated with each other by combining two patterns into one long concatenated vector. Each such vector is given as an input to the algorithm. Then the training procedure is the same as in the original SOINN algorithm, resulting in a clustering of all pairs in the combined input space. For recalling an associated pattern, one member of the originally trained pair is given as a key and SOIAM compares this vector with the corresponding part of every node in the network. A threshold criterion determines whether recall is successful for a node and, if successful, a representative node of the cluster which the node belongs to is added to a recall set of nodes. The algorithm has been successfully applied to learn one-to-one and one-to-many associations for both real-valued as well as bipolar input data.

Tangruamsub et al. (2012) applied SOIAM to the task of robot navigation in a simulated environment. However, the association of pairs was not performed by concatenating corresponding vectors and storing them in a combined input space. Instead, an additional associative layer is added on top of a memory layer. The memory layer employs a simplified version of SOINN and forms clusters within the data. These clusters can be paired with each other in the associative layer. Relationships between patterns can only be learned once the formation of the topology in the memory layer has finished. Moreover, the learning process requires a strict pre-specified order in which different inputs must be presented. In their experiment, a simulated robot had to associate the combination of an image taken from the robot's position and a specific movement action with the image obtained after performing this action. Based on these associations the system created a topological map that was used for path planning and allowed the robot to navigate to a desired goal position. Robot navigation based on a topological map of visual inputs was approached by Neal and Labrosse (2004). However, their approach did not include the association with movement actions and the topological map of visual inputs was created by an Artificial Immune System which is reviewed in Section 2.2.5.

SOINN was further employed in a model termed General Associative Memory (GAM) (Shen et al., 2013). GAM is divided into three layers, namely input layer, memory layer and associative layer. The input layer simply receives the data patterns. The memory layer consists of SOINN with only one layer, which learns a topology of the data and forms clusters of similar patterns. Finally, the associative layer builds relationships among these learned clusters. To learn an association between a key-response pair, the respective two inputs must be presented strictly in order, i.e. the key must be directly followed by the response. Repetitive presentation leads to an increment of the corresponding association weight. GAM uses information about the class of the current input, which means that the classes of the input patterns must be known and learning needs to be supervised. The GAM model can learn a temporal sequence of inputs by successively receiving each two subsequent patterns

as a key-response pair. No relationships are learned between non-adjacent inputs. Furthermore, GAM can perform pattern completion and thus cope with noisy inputs by performing the recall in an auto-associative way. Shen et al. (2013) evaluated GAM by learning and recalling both one-to-one and one-to-many associations as well as temporal sequences. GAM was tested on binary and real-valued data and in a practical task involving a robot. Also Yu et al. (2010) and Shen et al. (2010) described the same three-layer architecture but evaluated it on different datasets.

Some characteristics of SOINN can also be found in the Adaptive Resonance Theory. Corresponding similarities will be outlined in the following section.

2.2.4 Adaptive Resonance Theory

SOINN has similarities to the Adaptive Resonance Theory (ART) of learning (Tan et al., 2007; Grossberg, 2013). Both algorithms perform an unsupervised clustering of the inputs and work in an online and incremental manner. ART uses a so-called *vigilance* parameter to judge whether an input belongs to a certain category. Similarly, SOINN uses similarity thresholds for grouping inputs into clusters. In SOINN these thresholds are set per node and depend on the current topology. In ART only a single adjustable vigilance parameter exists per *input field* (Tan et al., 2007). ART represents categories with only one node (one weight/pattern), hence, learns hyperspherical clusters. SOINN represents clusters with multiple nodes (multiple weights/patterns) and can learn clusters of any shape. In ART the creation of a new node occurs when an input violates the vigilance criterion for all existing category fields. In SOINN a new node is created when a threshold value of the two closest nodes to the input is exceeded. However, in contrast to ART, a new node can be pruned again if no other similar inputs cause the formation of a cluster with this node.

ART has been applied in the Fusion ART architecture (Tan et al., 2007) which resembles the functionality of an associative learning architecture with multiple SOINN modules. The Fusion ART architecture uses multiple input fields and each of them captures another type of input (e.g. from different sensors). Each node of the category field stores an associative mapping over multiple input fields (Figure 2.3). Thus, one node represents exactly one mapping. When a specific pattern of one input field is associated with multiple patterns of another input field, just as many nodes are required to represent these associations. These associations do not have any modifiable strength values – either there is an association or not. GAM (Shen et al., 2013) follows a similar approach but can learn arbitrary associations which are both directed and weighted.

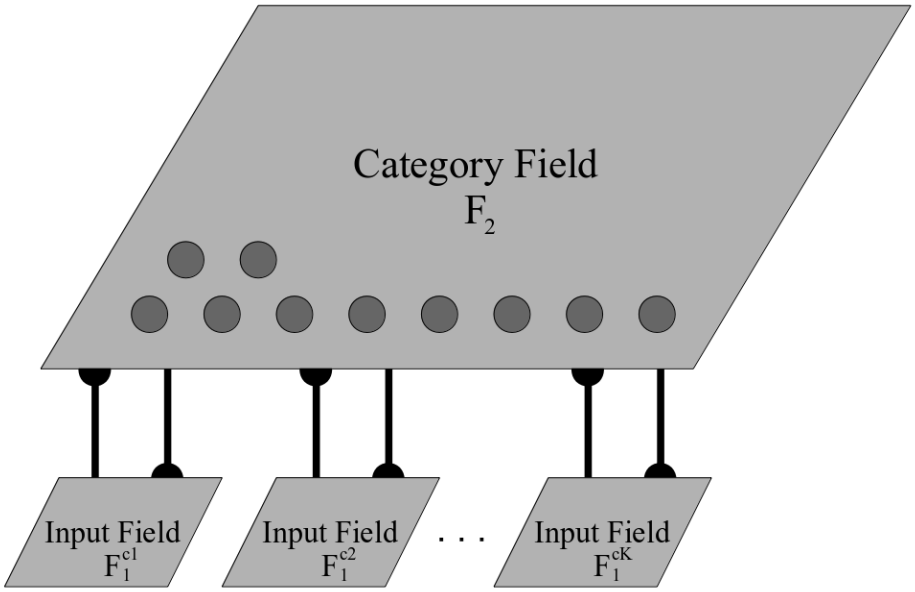


Figure 2.3: The Fusion ART architecture. The patterns of multiple input fields F_1^{c1} to F_1^{ck} are associated in the category field F_2. Adapted from Tan et al. (2007).

2.2.5 Artificial Immune System

An Artificial Immune System (AIS) is a model for unsupervised topology learning, inspired by the human immune system. AIS stores the received inputs in a topology of different *B cell objects*, each of which captures an input pattern (Timmis et al., 2000). The reception of an input results in a stimulation (excitation) of the B cell objects in the network. The level of stimulation depends on the closeness of the input to the stored perception and the distance to the perceptions of its neighbours (i.e. connected cell objects). Additionally, the neighbours suppress the stimulation of the respective cell object. If the stimulation level exceeds a specified threshold, then the corresponding cell is cloned and mutated. By choosing a low mutation rate, the pattern stored by the cloned cell object differs only in a few fields (dimensions) and will be similar to its parent's pattern. Cell objects are connected with links based on their distance and on the average distance of all connected objects in the network. In this way, the network captures relationships between cell objects and clusters can emerge. As more inputs are received in a specific region of the input space, cloning continues and variation is spread over more cell objects. Eventually, the suppression by the neighbours is strong enough such that no further cloning occurs. Finally, the weakest cell objects are removed from the network (Timmis et al., 2000).

A later variant of AIS, the Resource Limited Artificial Immune System (RLAIS), introduced the concept of *artificial recognition balls* (Timmis and Neal, 2001) with the purpose to limit the number of nodes stored in the network and, in this way, support continual learning. The recognition balls are hyperspheres spanned around input patterns and determine up to which range similar inputs are attracted. The *network affinity threshold* is fixed and sets the range of attraction for the whole

network. After each learning iteration the stimulus levels of all recognition balls are calculated and, based on these levels, the received inputs are allocated to recognition balls. If the number of assigned inputs is greater than the allowed maximum, inputs are removed from the recognition balls with the weakest stimulation. Any recognition balls without allocated inputs are removed entirely (Timmis and Neal, 2001). In the AIS algorithm used by Neal and Labrosse (2004), the resource levels of all recognition balls are stored persistently. These level are updated after each input based on the respective stimulation and a decay process. The removal of a recognition ball (culling) happens when the respective resource level falls below a specified threshold value (Neal and Labrosse, 2004).

A more comprehensive coverage of AIS, including details on the biological motivation behind, is given by de Castro and Timmis (2002, 2003).

2.3 Methods for Associative and Hebbian Learning

Associative learning becomes relevant for autonomous robots when no feedback about the structure of the incoming information is available or when the goal of learning is unclear.

Associative learning relates to the principles of conditioning where learning happens by associating perceived stimuli. Conditioning theory can be differentiated into classical conditioning and instrumental conditioning. Classical conditioning creates stimulus-stimulus-associations or stimulus-response-associations, while instrumental conditioning also trains to avoid responses with negative consequences by learning inhibitory associations (Anderson, 1999; Bear et al., 2006).

A popular and successful framework for classical conditioning is given by the Rescorla-Wagner theory (Rescorla and Wagner, 1972; Pearce and Bouton, 2001). A fundamental part of this theory is that learning is proportional to the difference between the current association strength and the respectively achievable maximum strength (Anderson, 1999) – resulting in weaker learning for stronger associations. This rule can be extended to a principle for prediction error correction by comparing the desired activation of a node with the activation predicted via its associations from other stimuli. Such a mechanism has been applied in various forms to other models of classical conditioning (Balkenius and Morén, 1998). The Rescorla-Wagner theory states further that learning is inversely proportional to the number of items to be associated (Anderson, 1999). In other words, stimuli compete for associative strength. Wagner (1981) proposed two different activation states of a node (termed *A1* and *A2*) to distinguish between the activation caused by perception and the activation caused by spreading. Learning only occurs when a stimulus is present (in *A1* state) whereby the state of the response node determines whether the influence

is exhibitory or inhibitory. Modifications on the influence of the different state combinations have been suggested by other researchers but it remains controversial how the learning of exhibitory and inhibitory associations should be influenced by these states (Pearce and Bouton, 2001).

Classical conditioning theory clearly distinguishes between conditioned and unconditioned stimuli. Such a distinction is not made in ISO-learning suggested by Porr and Wörgötter (2003) where all inputs are treated equally. However, instead of allowing to learn an association between any two inputs, Porr and Wörgötter (2003) predefined inputs and outputs. Then only the connection from a respective stimulus to a predefined response is learned such that the learned stimulus augments an existing reflex stimulus.

A popular model of associative memory was introduced by Hopfield (Hopfield, 1982, 1984; Haykin, 2008). In the Hopfield model binary threshold units represent both inputs and outputs. These units are fully connected in a network without self-connectivity. Learning is accomplished through Hebbian learning with the option of negative weight changes. A limitation of the Hopfield model is the predetermined fixed size of the network. The network cannot grow incrementally and, thus, the model is limited in the amount of information it can represent.

2.3.1 Multimodal Associative Learning Architectures

Learning relationships between signals from different sensory-motor modules is performed by the XCR-1 robot, an experimental robot empowered by the Haikonen Cognitive Architecture (Haikonen, 2011). The Haikonen Cognitive Architecture is a neural architecture that is able to use sub-symbolic signal patterns as symbols. The sensory input undergoes an informed preprocessing transformation and the resulting features then form associations with features obtained from other modules.

Associating data of different modalities is performed by the Distributed Associative Interactive Memory (DAIM) architecture introduced by Baxter et al. (2013). DAIM uses a Hebbian learning rule with a saturation term and the flow of activation between units serves for inference purposes. Despite being used in the context of human-robot interaction, DAIM on its own cannot cope with data as is read from the robot's sensors but requires separate preprocessing steps. For instance, Baxter et al. (2013) manually discretize numeric values before passing the discretized data on to DAIM.

Another multimodal architecture was proposed by Vavrečka and Farkaš (2014) for unsupervised symbol grounding. This connectionist architecture links the inputs of different modalities to abstract symbols. Vavrečka and Farkaš (2014) applied methods for dimensionality reduction and topology formation in order to represent different perceptual concepts.

The idea of storing symbolic information within a neuron has been introduced by

Velik and Bruckner (2008), who termed corresponding units *neuro-symbols*. Neuro-symbolic networks made up of these units are intended to bridge the gap between distributed information storage found in neural networks and symbolic information processing systems. Velik (2007) suggests a hierarchy of neuro-symbols with lower levels capturing sensory information and higher levels representing more abstract concepts.

2.3.2 Associative Learning with Self-Organizing Maps

Based on SOM, various approaches have been proposed that combine topology learning with a mechanism to model temporal context or some sort of Hebbian learning.

In the Recurrent Self-Organizing Map (RSOM) (Varsta and Heikkonen, 1997; Koskela et al., 1998) SOM is extended with a memory for previous inputs. Each unit has a so-called *difference vector* assigned to it, which stores a weighted sum of its previous value and the distance of the corresponding weight vector to the current input. This variable acts as a leaky integrator of differences to received inputs. These difference vectors are also used for determining the best matching unit for each input. After having updated the difference vectors for a given input, the unit having the shortest difference vector is declared the winning unit. In addition, the subsequent updates of the weight vectors of winning unit and neighbours are based on the corresponding difference vectors (instead of using the distance to the input as is done in the standard update rule for SOM). By keeping information about previous inputs, RSOM models sensitivity to context.

In another approach termed Merge-Self-Organizing Map (MSOM) (Strickert and Hammer, 2003) context is represented explicitly as a linear combination of the weight vector and the context vector of the best matching unit in the previous step. For determining the best matching unit to a given input, both the spatial distance (respective weight vector to the current input) and context distance are taken into account. Such a context measure can also be incorporated into the Neural Gas algorithm (Martinetz et al., 1993), resulting in the Merge-Neural Gas (MNG) algorithm (Strickert and Hammer, 2003).

Koutnik introduced the Temporal Hebbian Self-Organizing Map (THSOM) (Koutnik, 2007; Koutník and Šnorek, 2008) which extends SOM with recurrent connectivity (Figure 2.4). Every unit is connected to every other unit as well as to itself with trainable, directed edges. These connections are named the *temporal map* whereas the weight vectors stored in all the units are referred to as *spatial map*. Additionally, each unit stores an activation value which is computed based on the similarity to the current input (measured by the distance to the corresponding weight vector) and the activation received from other units. The influence of each of these two parts can be regulated with a parameter. After each update all activation values

are normalised such that the highest activation value corresponds to 1. A modified Hebbian learning rule is used to evolve the connection weights in the temporal map. The unit with the highest activation increases its connection weights to all other units. All remaining weights suffer an exponential decay dependent on the current weight value. THSOM uses a network of a fixed size and the number of units must be determined in advance.

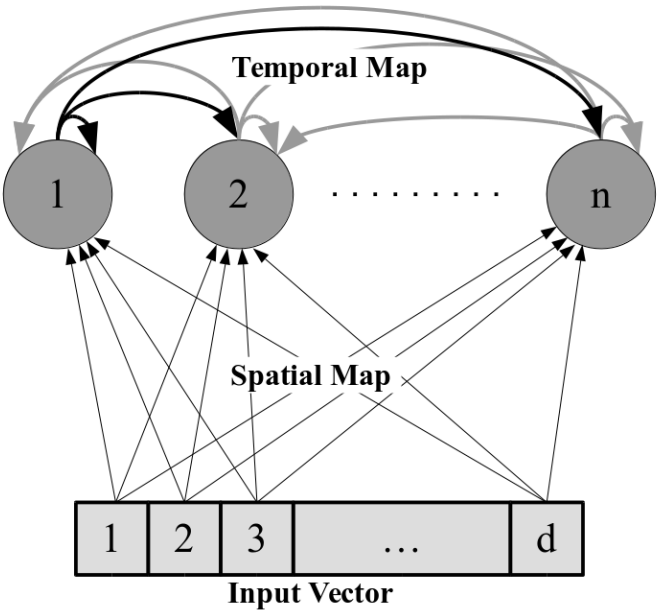


Figure 2.4: The THSOM architecture with a d-dimensional input vector and a network layer with n units. The spatial map stores the topological structure of the inputs and the temporal map encodes the temporal relationship of different units. Adapted from Koutník and Šnorek (2008).

Based on THSOM, the Temporal Network for Transitions (TNT) (Graziano et al., 2011) is a SOM architecture which also takes into account the action taken by the network and thus incorporates a form of reinforcement learning. Like THSOM, TNT uses spatial activation (depending on the current input) and temporal activation (depending on the previous network state) to determine the new state of the network as well as to drive learning. TNT combines temporal and spatial activation multiplicatively and can achieve better results than with an additive combination (Graziano et al., 2011). Instead of utilising only one single temporal map, in TNT separate matrices (called *transition maps*) can influence the spreading of activation differently and are trained separately. To ensure convergence to a stable state, SOM decreases parameters such as learning rate and neighbourhood size during learning. These parameters are used globally in the whole network. To each of its units, TNT assigns a local age value which can decrease independently of the others. Learning rate and neighbourhood size can be set independently for different units which means that sensitivity to new inputs and, in turn, plasticity can vary in different network regions. Graziano et al. (2011) report that TNT performs better on larger

input spaces and can better cope with noise, compared to both SOM and THSOM. However, TNT also suffers from the drawback of being static in its size and requires the number of units to be decided in advance.

2.3.3 Sequence Learning and Serial Order Recall

By concatenating associations between numerous pairs of temporally offset stimuli, sequential information can be modelled. In the field of psychology, sequence learning has been studied in the context of serial order recall.

Lewandowsky and Murdock Jr. (1989) proposed a memory model for serial order as an extension of Murdock's Theory of Distributed Associative Memory (TODAM) (Murdock Jr., 1983). A major assumption of TODAM is that serial recall is based on the associative chaining of items, a concept first introduced by Ebbinghaus (1913). Each item is represented as a multi-dimensional normalised random vector. In order to learn an association from an item A to an item B, the vector of A is convolved with the vector of B. The result is then added to a memory vector that holds all learned associations. When storing multiple associations in sequential order, a forgetting parameter causes later sequence items to be stored more weakly. This parameter can be omitted when using a similarity measure between item and memory vector to determine the encoding strength for this item. Additionally, the memory vector needs to store information for each single item. Hence, once an item is learned, the corresponding item vector is added to the memory vector. The encoding strengths for item and order information can be controlled by two weighting parameters. For recalling an associated item, the vector of a cue item needs to be correlated with the memory vector and the resulting vector further needs to be de-blurred to retrieve the original item vector. Serial recall happens by consecutively executing these steps for several items – starting with the vector of the first sequence item and always using the last retrieved vector as a cue for the next item. With their model Lewandowsky and Murdock Jr. (1989) aim to reproduce empirically established serial position curves and memory span functions and to simulate partial and delayed recall effects. Although the model succeeds in fitting corresponding empirical data, it requires explicit information about the current list position, which is used to adjust the weighting parameters during learning. Also the size of the memory vector was manually chosen according to the number of items competing for recall.

Solway et al. (2012) examined the effects of positional clustering and temporal clustering with an associative chaining model. Positional clustering in serial recall refers to the observation that subjects are more likely to recall items close to their absolute list positions, even after making serial order errors. Temporal clustering describes the finding that items are more likely to be recalled close to their neighbouring items, i.e. items that were adjacent in the studied list. In general, the latter effect is better predicted by models based on associative chaining, whereas

positional clustering is better accounted for by models that make use of explicit position-to-item associations. The associative chaining model described by Solway et al. (2012) successfully reproduces both effects and partially provides a better fit to the empirical data than a popular model of Burgess and Hitch (discussed below) which relies on positional coding. Based on these results, Solway et al. (2012) conclude that, apart from temporal clustering, also the positional clustering effect is primarily caused by a contiguity-based associative mechanism.

Burgess and Hitch (1999) proposed a connectionist model for verbal serial recall. The model uses localist representations for the different sequence items and distributed representations for phenomes that serve as input and output signals. Additionally, context is modelled in a distributed fashion and acts as another input during both learning and recall. The item nodes bind the varying context/timing signal to specific phenomes. Connections exist from context nodes to item nodes, from input phenome nodes to item nodes, and from item nodes to output phenome nodes. Corresponding connection weights are modifiable and can be learned in a Hebbian manner (Hebb, 1949). Moreover, input and output phenome nodes are linked with hard-wired one-to-one connections which reproduce the activation of the input phenomes at the output phenomes and vice versa. The resulting feedback between verbal input and output is inspired by the concept of the phonological loop, which was suggested by Baddeley (1986). The model successfully simulates primacy and recency effects as observed in serial position curves and predicts order errors as well as serial order intrusion. Also considered are factors such as list length, word length, Hebb repetition, phonetic similarity and temporal grouping.

Burgess and Hitch (2006) proposed a revised model that can learn multiple sequences simultaneously and cope with partially repeated lists. This is achieved by allowing multiple sets of context nodes. During learning, the model tries to match the presented sequence with each existing set of context nodes. The best matching set is used for learning the context-to-item associations. If no good match is found with any of the existing sets, a new set of context nodes is dynamically created. Thus, the model can learn an arbitrary number of sequences as long as these sequences are distinct enough to each other. In the revised model serial intrusion errors occur because of the erroneous matching of a sequence with an older incorrect context set. The matching mechanism also provides an explanation for proactive interference.

An important assumption of both models is that context is responsible for producing serial recall, instead of being caused by associative chaining. During both learning and recall, the current list position is reflected in the pattern of active context nodes. While progressing through the list, a moving window of activation moves over the context nodes and results in a continuously changing context. During learning, the corresponding context-to-item associations are established; and during recall, the context nodes activate associated item nodes. A winner-takes-all selection

process and response suppression support the recall of one item after the other. However, the moving window needs to be manually set and must be externally controlled. In this regard, the actual serial order of the presented items is not learned by the model itself but explicitly given through the context. The model simply learns associations between two sets of nodes but cannot switch to associations for subsequent items by itself. The list progression along serial order is delegated to an external control mechanism. Such a mechanism could be provided by continuously changing oscillatory patterns but it still remains unclear how the context signal can be reset to the correct starting pattern when initiating a later recall process.

Another distributed model for serial recall is the "serial-order-in-a-box" (SOB) model, introduced by Farrell and Lewandowsky (2002). The SOB model uses binary feature vectors to represent item and an auto-associative network in one layer to represent associations. A matrix stores excitatory weights that can be strengthened by Hebbian learning. Additionally, another matrix accumulates inhibitory weights during recall to enable response suppression. For recalling a sequence of items, the network is cued with a very short random vector, and the dot product of (the sum of) the weight matrices with the cue vector is computed, resulting in the initial network state. Then the dot product with this vector is computed and combined with the current network state to obtain the new network state. This process is repeated until the network converges towards an attractor state, which represents the current response of the network. After having output the response, the process starts again for the next sequence item. To suppress given responses in future recall processes, the negatively weighted auto-association of the most recent response vector is added to the matrix of inhibitory associations. In other words, the SOB model consists of a fully-connected network of units and associations and each association stores both an excitatory and an inhibitory weight. Hebbian learning between all units changes the excitatory weights. During recall, all units are randomly initialised with low activations and activation spreads over all associations, influenced by both excitatory and inhibitory weights. Spreading continues until an attractor state is reached, then the current activation state is returned as a response. Hebbian learning is used to (negatively) update all inhibitory weights, and the entire process is performed again for obtaining the next response. For short lists of five items the model successfully replicates serial position curves and phenomena such as repetition errors, list length effects and word frequency effects. The primacy gradient is modelled by using the energy of the network to influence the encoding strength of the current item.

Also Botvinick and Plaut (2006) used a recurrent neural network to model serial order recall. The entire network consists of three layers: an input, a hidden and an output layer. Connections exist from input to hidden layer, within the hidden layer, as well as from hidden to output layer and vice versa. A localist representation is used for the different items, i.e. an item is represented by a single unit in the input

layer and a single unit in the output layer. For computing a unit's activation in the hidden layer, the weighted sum of inputs is passed to the logistic function; and the softmax function is used to obtain the activations in the output layer (Rumelhart et al., 1995; Haykin, 2008). The network learns by using a form of backpropagation through time (Williams and Zipser, 1995). While learning a sequence, the units in the input layer were activated one by one and the activation values of the output units were fixed at the desired target values. Learning continued until the model reached a certain level of recall accuracy. For recalling the sequence, only the cue unit was activated in the input layer and remained active until the end of the recall phase. Botvinick and Plaut (2006) tested their model against various phenomena of serial recall. The curve of the model's recall accuracy shows a sigmoidal shape with increasing list length, which is in line with empirical findings reported by Crannell and Parrish (1957). The model accounts for the primacy effect and weakly for the recency effect. Moreover, the model predicts fill-in errors and infrequently occurring relative errors as well as effects of item similarity and the sawtooth pattern for lists with alternating confusable and non-confusable items. Furthermore, Botvinick and Plaut (2006) investigated artificial grammar learning on lists with repeated list items, where their model exhibits effects of familiarity with the underlying regularities.

Unintentionally and unconsciously learning regularities which are contained in the perceived input stream is a form of implicit learning (Cleeremans and Dienes, 2008). In this regard, learning occurs whenever information processing takes place (Boyer et al., 2005). The learned but unconscious knowledge can be tested with experiments on sequence learning and artificial grammar learning. In both scenarios the presented material has an underlying structure and subjects learn these regularities by repeatedly being exposed to the material, without being informed of any of the structural properties. Artificial grammar learning focusses on the recognition performance when distinguishing grammatical from non-grammatical strings whereas sequence learning in the domain of implicit learning observes reaction times when predicting the next element in previously learned sequential material (Cleeremans et al., 1998).

An influential model of implicit learning was proposed by Elman (1990) who used a feed-forward neural network with recurrent connections to context units, which includes information about previous states. The network was tested on sequences of letters in order to solve the task of artificial grammar learning. However, the model required an explicit distinction between current and succeeding input. In contrary, a model examined by Dienes (1992) utilises an auto-associator network with only one layer of units for both input and output. Dienes (1992) applied either the Hebb rule or the Delta rule for learning the sequential structure of binary encoded letter features and found the Delta rule to be more suitable for artificial grammar learning. Dienes (1992) either presented the grammar strings as a whole, entire

patterns or successively letter by letter. The model of Dienes (1992) can learn the regularities between different letters but was not tested on recalling letters in a sequential order. In the case of presenting the strings successively in a sequential order, the network contained information about the position of the letters in its (sequentially feed-forward connected) structure. Hence, it did not learn anything about the sequential order arising from the presentation of inputs. The associative memory model proposed in Chapter 3 is able to learn the sequential order of inputs without requiring any predefined structure.

Learning asymmetric relationships between different stimuli can be achieved by utilising the temporal derivatives of the stimulus signals. This is done in differential Hebbian learning (Kosko, 1986; Wörgötter and Porr, 2005), a learning method employed by models of Sutton and Barto (1981, 1990), Klopf (1988) and Balkenius (Balkenius and Morén, 1998).

The use of directed associations for representing the relationship between two stimulus items is postulated by the independent association hypothesis (IAH), which allows different weight values to develop independently for the two possible directions. In contrary, the associative symmetry hypothesis (ASH) (Kahana, 2002) considers only one single associative strength for both forward and backward association between two items. Although various studies, mostly on paired-associate learning, deliver support for the ASH, corresponding results do not allow to rule out the IAH (Rizzuto and Kahana, 2000, 2001; Kahana, 2002). On the other hand, evidence for the IAH is provided by studies involving longer serial lists of items (Li and Lewandowsky, 1995; Kahana, 1996; Rosen and Engle, 1997; Kahana and Caplan, 2002). Obviously symmetric connection weights can easily emerge using unidirectional edges but asymmetric associations cannot be developed under the ASH by definition. However, under certain conditions, namely when the network topology biases recall towards one of the associated items, models based on the ASH can produce asymmetric retrieval effects (Kahana, 2002).

2.4 Summary

This chapter elucidated how the research described in this thesis links and relates to various published works in the fields of machine learning and psychology. Several topology learning approaches were described with the focus on the GNG and SOINN algorithms as these approaches support incremental clustering without restricting the topology layer to a fixed size. Also relevant research in psychology was identified, including studies on associative learning and serial order recall. Those empirical findings and theories provided a basis for the design and modelling decisions made for the proposed architecture, which will be explained and evaluated in the following chapters.

Chapter 3

ICALA: Incremental Clustering & Associative Learning Architecture

For achieving the goals set in Section 1.3, a learning and memory architecture has been developed, which is modular and capable of handling perceptions from multiple sensors and controlling different actuators. This chapter first describes the structure of the entire architecture and illustrates its general functionality. Then the two major components are explained in depth, including corresponding algorithms and equations. Finally, details about the integration of both components are given.[1]

The developed architecture is based on two central parts, one being an associative memory model which learns and stores relationships between different input categories. The other part is an incremental topology learning algorithm which is able to extract categories from perceived input data. The topology learning algorithm can be repeatedly incorporated into various modules. The proposed architecture reads multi-dimensional data from various sensors. While receiving inputs, the architecture incrementally builds a topology in which similar inputs are grouped together and clusters emerge. These clusters represent abstract categories found in the input data. Using these clusters, the architecture learns relationships between different input categories by forming associations based on the co-occurrence (and timing) of the inputs. With these associations the architecture can recall previously learned relationships. The recalled data can then be used together with the multi-dimensional data stored in the topology to compute an output vector, which, in turn, can affect an actuator. This process enables the architecture to learn and execute different stimulus-response behaviours.

Methods for clustering are combined with associative learning techniques, and thus make it possible to learn relationships between different distinct data patterns while allowing the direct use of sensory data as input. Any perceived data are first passed to a topology learning algorithm, which incrementally learns a cluster

[1] A Java implementation of the entire architecture is available at `https://github.com/MatthiasKeysermann/ICALA`.

structure. The emerging clusters are the union of multiple similar inputs. Therefore, the cluster representation becomes tolerant to small variations and the algorithm can generalise over several similar inputs. When receiving data, the current input gets assigned to one cluster or otherwise forms a new cluster. This cluster is considered as the currently active cluster. In order to learn associations between different clusters, data from various modalities must be incorporated. This was achieved by designing the desired learning architecture in a modular way – for each possible data source a separate module can exist that independently learns its own network topology. Whenever inputs from different modalities enter the architecture simultaneously, multiple clusters become active at the same time. The information about all currently active clusters is passed to an associative memory model that forms associations between these clusters. In this regard, the topology learning algorithm separates the input data in their original input space and performs spatial clustering. The associative memory model performs temporal clustering as it associates inputs based on their co-occurrence. Thus, the data is grouped based on both spatial similarity and temporal closeness.

The topology learning algorithm is implemented as a modified version of SOINN, named the Modified Self-Organizing Incremental Neural Network (M-SOINN), which will be described in Section 3.1. The associative memory is implemented in a model termed Temporal-Order Sensitive Associative Memory (TOSAM). Details about TOSAM will be given in Section 3.2. In the following the terms *node*, *edge* and *cluster* refer to corresponding entities in M-SOINN, whereas *unit* and *connection* or *association* stand for information units and associations between them in TOSAM.

Figure 3.1 shows the integration of both methods into a larger architecture, the Incremental Clustering & Associative Learning Architecture (ICALA). Data can originate from various sensors and is read by the corresponding M-SOINN modules. In each module M-SOINN builds a topology of the inputs. Currently active clusters serve as inputs for TOSAM, which creates a unit for each received cluster input. Based on the co-occurrence of these cluster inputs, TOSAM learns associations between corresponding units. By propagating activation over these associations, TOSAM can activate connected units. This enables the recall of previously associated information, i.e. when only one of the associated units gets activated as a recall cue. Furthermore, each M-SOINN module can read the activation levels of the units corresponding to its own clusters. The module can then combine these activation levels to form a weighted sum of the corresponding cluster centroids. The result is a new instance in the original input space that can be used as an output, e.g. to manipulate an actuator.

If such an actuator operates in an equivalent space as the assigned sensor, the generation of the output is based on sensor readings and can be directly used to control the actuator (e.g. a robotic arm that can sense its current position). This

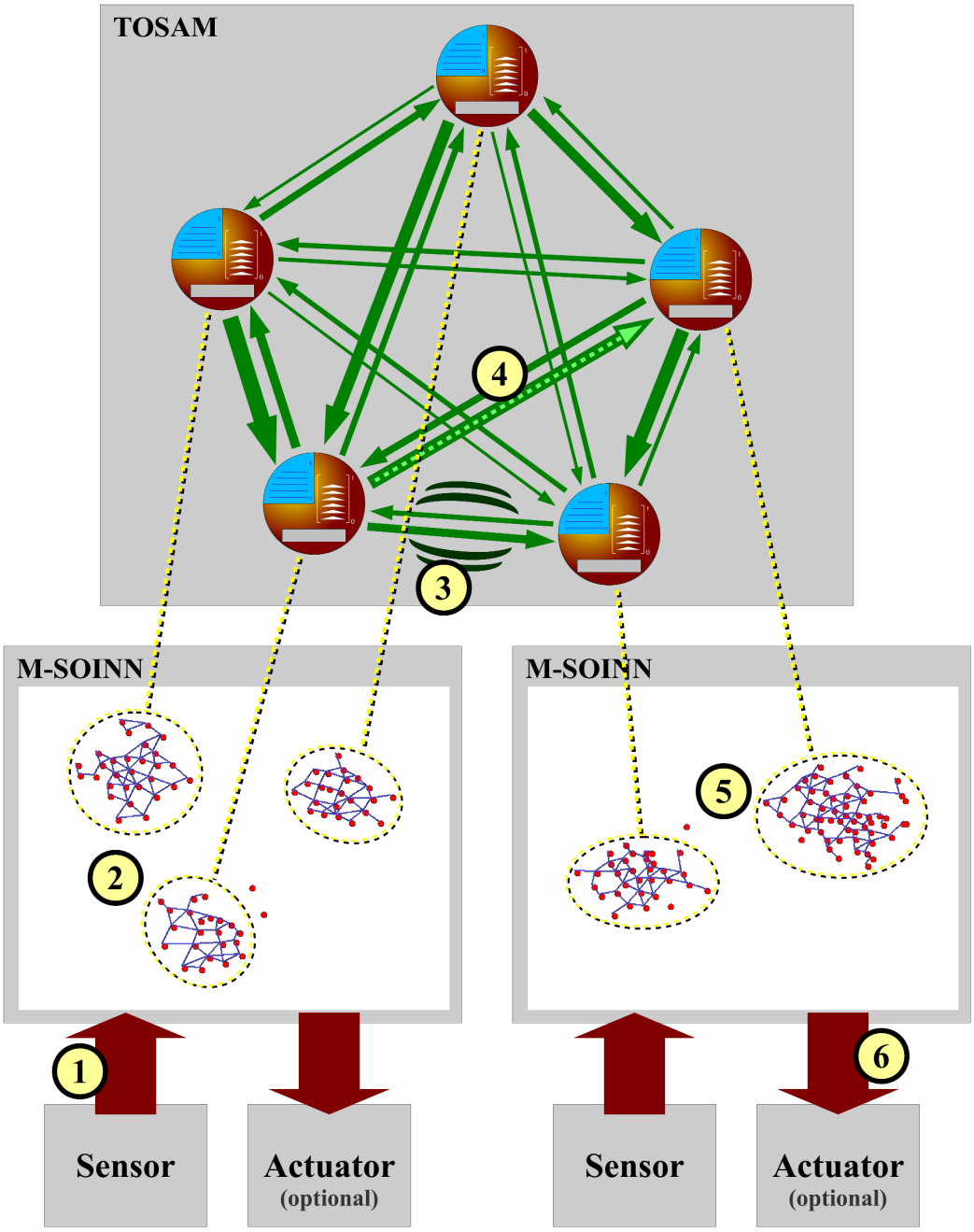


Figure 3.1: TOSAM and M-SOINN modules integrated into ICALA. An M-SOINN module reads a data vector from a sensor (1). In an M-SOINN module a topology of the inputs is learned and the data is clustered (2). Each cluster corresponds to a unit in TOSAM. Upon reception of an input into a certain cluster of an M-SOINN module, the corresponding unit in TOSAM gets activated. TOSAM learns and strengthens the associations between each pair of activated units (3). Based on the associative strengths the units spread their activation and activate other units (4). The activation levels of all the units corresponding to a module's clusters are passed back to the respective module. These levels and corresponding cluster centroids are combined to produce a data vector (5), which can then be used to influence an actuator (6).

implies that both sensor and actuator work with the same data – or at least with the same type of data. To clarify what is meant with an equivalent input-output space, a few examples are given. For instance, both a camera and a screen operate on visual data and use a planar pixel array to represent the data. In this case, the dimensionality of input and output can even differ as the data samples can be scaled to a common dimensionality. Also a microphone and a speaker share the same input-output space. Both deal with audio data that can be represented as a multi-dimensional vector of frequency intensities in the Fourier spectrum. Even a robot arm can provide current joint angle readings as an input and afterwards set these angles according to the calculated output.

Each M-SOINN module runs at a certain frequency at which it always receives sensory data as input and sends output data to control the respective actuator (if present). These frequencies can be set differently for each module and are independent of the frequency at which TOSAM runs. This allows ICALA to adapt to different sensor/actuator types and provides more flexibility when handling the data of different modalities.

Furthermore, the activation of units in TOSAM can be restricted such that the inputs into smaller clusters do not result in an activation. Only clusters of a minimum size are passed on to TOSAM and influence learning. A minimum number of nodes can be set as a threshold value. In this way, only clusters that contain a minimum number of nodes cause the activation of their corresponding units in TOSAM. This process filters out inputs that are stored in the topology but may be only temporary and possibly not very important in the long run. Only perceiving several similar inputs leads to the creation of bigger clusters in the topology. Such clusters contain more inputs and can be recognised more easily in the future. They can be considered as more representative and are therefore worth including for associative learning, in contrast to very small clusters.

In the following, first the M-SOINN algorithm is explained, then the TOSAM model is described.

3.1 M-SOINN: Modified Self-Organizing Incremental Neural Network

The M-SOINN algorithm is based on the SOINN algorithm and inherits parts from GNG and E-SOINN. But M-SOINN alters some processing steps and introduces new operations for improving the topology. The algorithm sequentially processes inputs and incrementally forms a topology of nodes and edges. New inputs can either result in new nodes or lead to an adaptation in the respective local topology. Global refinement operations are performed in regular intervals. Figure 3.2 provides a visualization of all processing steps, which are explained in the following. The

complete M-SOINN algorithm is listed in Section 3.1.3.

In contrast to SOINN, M-SOINN learns the topological structure in only one layer. Any second-layer learning would interrupt first-layer learning and cause the architecture to be unresponsive to new inputs during that period. Instead of initialising the network with two random patterns as it is done in GNG and SOINN, the network of M-SOINN is empty at the beginning. In this way, M-SOINN avoids any biases generated by a random initialisation. Upon reception of an input, whenever there are less than two nodes in the network, the current input becomes a new node. In this case no other processing steps take place until the next input is received. Thus, M-SOINN can cope with an empty network at any time. If the network contains at least two nodes then the following processing steps are executed.

3.1.1 Input Processing

The algorithm determines the two nodes nearest to the current input. For these two nodes *similarity thresholds* are calculated based on the maximum (Euclidean) distance to their respective neighbours (directly connected nodes); or the minimum (Euclidean) distance to other nodes in case the respective node has no neighbours. If M-SOINN processes many highly similar inputs in a row, the similarity thresholds of the created nodes can quickly become very small. Such threshold values do not allow the formation of bigger clusters. Thus, M-SOINN allows setting the similarity thresholds to a predefined fixed value. This avoids the creation of numerous very small clusters and mitigates the increase in the number of nodes. Fixed similarity thresholds are also employed in other approaches based on GNG (Marsland et al., 2002; Prudent and Ennaji, 2005).

Furthermore, the Euclidean distances from the input to each of these nodes are calculated. If any of these distances exceeds the respective similarity threshold, the algorithm creates a new node at the position of the input. A reasonable modification of creating edges based on similarity thresholds is given by Prudent and Ennaji (2005). In their variant of GNG, an edge can be created if the received input is similar enough only to the nearest node. In this case, the newly created node is directly connected to its nearest node. This process tends to extend existing clusters while still allowing dissimilar inputs to remain isolated. M-SOINN adopts these processing steps for connecting new nodes but adds two restrictions. First, an edge is only created if the new node would extend the cluster of the nearest node. In other words, the new node must be further away from the cluster mean than the determined nearest node. Second, the distance between the new node and the nearest node must be greater than the average node distance in the corresponding cluster. This avoids inflating the cluster with many nodes very close to each other.

If none of the above-mentioned Euclidean distances exceed corresponding similarity thresholds, the algorithm creates an edge between the two nearest nodes (if such

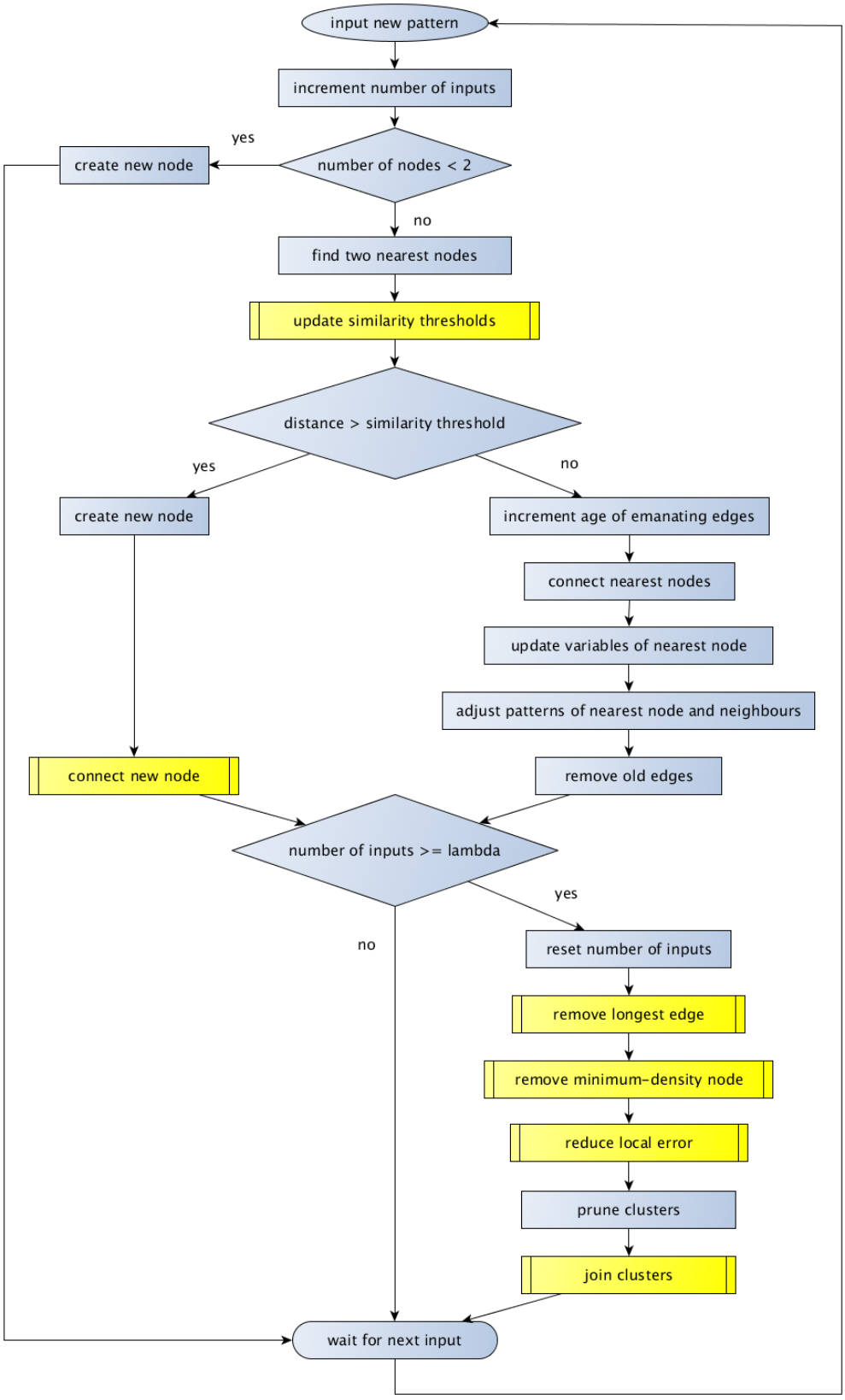


Figure 3.2: Processing steps of the M-SOINN algorithm when a new input is received. The introduced modifications are shown in yellow boxes with double lines. A complete description of every single execution step can be found in Section 3.1.3.

an edge did not exist before) and resets the corresponding *age* value to 0. Then the age values of all edges emanating from the nearest node are incremented. For the nearest node the *local accumulated error* is increased by the Euclidean distance to the current input. Additionally, a counter is incremented that stores the number of times this node has been nearest node, denoted as the *number of signals*. Afterwards M-SOINN adjusts the positions of both the node itself and its neighbours to be closer to the current input. Then the algorithm removes edges whose age is greater than a pre-specified threshold age_{dead}.

3.1.2 Topology Refinement

In regular intervals, after a fixed number of inputs λ, various steps are executed to tidy up the learned topology as it is done in both GNG and SOINN. In M-SOINN this clean-up process uses processing steps from both GNG and SOINN but also adds new measures to improve the topology. Any of the modifications introduced in M-SOINN can be separately deactivated.

One addition of M-SOINN is a removal criterion for edges. After having determined the edge with the maximum length, this edge is removed independently of any other criteria. As the chosen edge is the longest in the network, the relationship between the two connected inputs can be expected to be weaker than for any other connected inputs. This edge could be the connection between two clusters that form – due to this connection – one joint cluster. The removal of the edge allows for splitting up this joint cluster and potentially leads to a more accurate topology.

Another addition is the removal of the node with the minimum number of signals. The number of signals represents the input density in a given region. The node with the lowest density in the network is likely to be in a low-density region. Such a node is not very representative for its actual input, i.e. the deviation from the cluster centroid is expected to be relatively high. Removing such nodes facilitates a better separation of clusters and, hence, a cleaner topology.

Inherited from GNG is a node insertion process for reducing the local error in the region of the network with the highest error. The algorithm determines the network node with the highest error as well as the maximum-error node amongst the connected nodes. A new node is created halfway between these two nodes, is connected to both of them and the edge between these two nodes is removed. This insertion is performed unconditionally and independently of any error reduction conditions.

Adapted from E-SOINN is the removal of nodes with only a few neighbours. M-SOINN prunes clusters by removing nodes with no more than two neighbours. While isolated nodes are always removed during this step, the removal of nodes with one or two neighbours depends on a threshold criterion based on the number of signals of the node in question and the average number of signals over all nodes in

the network.

M-SOINN is able to join clusters located close to each other. Such clusters could possibly represent the same input category when a single cluster for all these inputs would be more appropriate. For joining two clusters the minimum distance between two nodes of any different clusters is determined. For those two clusters also the average distance for all node pairs within each of these clusters is computed. In case the distance between the clusters is smaller than each cluster's average node distance multiplied by a specified tolerance value, an edge is created in order to join these two clusters. However, for small clusters with many nodes close together, the average node distance is relatively small, which mostly disqualifies these clusters for being joined. Instead of using a relative join tolerance value, the distance calculation can also be based on an absolute join tolerance value. An absolute value allows clusters with a high node density to be joined.

3.1.3 Complete Algorithm

This section describes the complete M-SOINN algorithm. The following annotations are used:

- $||\cdot||$ denotes the Euclidean distance measure.
- dim denotes the dimensionality of the input patterns.
- p denotes the current input pattern.
- $count_p$ denotes to number of inputs since the last clean-up step.
- N denotes the set of nodes in the topology.
- thr_n denotes the similarity threshold for node n.
- err_n denotes the accumulated error for node n.
- sig_n denotes the number of signals for node n.
- E denotes the set of edges in the topology.
- age_e denotes the age of edge e.
- (n, m) denotes the edge between node n and node m and is equivalent to (m, n).
- $neighbours(n)$ denotes all nodes that are directly connected to node n with an edge, i.e. $m \in neighbours(n) \iff (n, m) \in E$.
- C denotes the set of clusters in the topology. Each cluster c has a set of nodes assigned to it and N_c denotes the set of nodes assigned to cluster c.

The steps for processing an input and for the refinement of the topology are illustrated in the following:

1. Input a new pattern $p \in \mathbb{R}^{dim}$.
2. Increment the number of inputs: $count_p = count_p + 1$.
3. If the number of nodes is less than two ($|N| < 2$):
   (a) Create a new node n_{new} with pattern p: $N = N \cup n_{new}$ with $p_{new} = p$.
   (b) Initialise the number of signals for node n_{new} to 1: $sig_{n_{new}} = 1$.
   (c) Initialise the accumulated error for node n_{new} to 0: $err_{n_{new}} = 0$.
   (d) Create a new cluster c_{new}: $C = C \cup c_{new}$.
   (e) Assign node n_{new} to cluster c_{new}: $N_{c_{new}} = N_{c_{new}} \cup n_{new}$.
   (f) Proceed with the next input (Step 1).
4. Find the two nodes which are nearest to the current input pattern p:
   (a) $s_1 = \arg\min_{n \in N} ||p - p_n||$
   (b) $s_2 = \arg\min_{n \in N \setminus s_1} ||p - p_n||$
5. Update the similarity threshold thr_s for node s $\forall s \in s_1, s_2$:
   - If a fixed threshold is used, set the threshold according to the parameter τ: $thr_s = \sqrt{\tau^2 \cdot dim}$.
   - If a dynamic threshold is used:
     – If node s has neighbours (edges exist between node s and other nodes), set the threshold to the maximum distance to its neighbours: $thr_s = \max_{n \in neighbours(s)} ||p_s - p_n||$.
     – If node s is isolated (no edges exist between node s and other nodes), set the threshold to the minimum distance to any other nodes: $thr_s = \min_{n \in N \setminus s} ||p_s - p_n||$.
6. If at least one of the distances to the two nearest nodes exceeds the corresponding similarity threshold ($||p - p_{s_1}|| > thr_{s_1}$ or $||p - p_{s_2}|| > thr_{s_2}$):
   (a) Create a new node n_{new} with pattern p: $N = N \cup n_{new}$ with $p_{new} = p$.
   (b) Initialise the number of signals for node n_{new} to 1: $sig_{n_{new}} = 1$.
   (c) Initialise the accumulated error for node n_{new} to 0: $err_{n_{new}} = 0$.
   (d) Create a new cluster c_{new}: $C = C \cup c_{new}$.
   (e) Assign node n_{new} to cluster c_{new}: $N_{c_{new}} = N_{c_{new}} \cup n_{new}$.

(f) *(Connection of New Node)*
If the distance to the nearest node is within the threshold of the nearest node ($||p - p_{s_1}|| \leq thr_{s_1}$):

i. Let c denote the cluster which node s_1 belongs to.

ii. Check if the distance to the cluster mean is greater for node n_{new} than for node s_1:

A. Let $\overline{p_c}$ denote the mean of cluster c with $\overline{p_c} = \frac{\sum_{n \in N_c} p_n}{|N_c|}$.

B. Check if $||\overline{p_c} - p|| > ||\overline{p_c} - p_{s_1}||$ (condition 1).

iii. Check if the distance between node n_{new} and node s_1 is greater than the average distance between nodes of cluster c:

A. Let $\overline{d_c}$ denote the average distance between the nodes of cluster c.

B. Check if $||p - p_{s_1}|| > \overline{d_c}$ (condition 2).

iv. If both conditions 1 and 2 are fulfilled:

A. Create an edge between node n_{new} and node s_1:
$E = E \cup (n_{new}, s_1)$.

B. Execute *(Cluster Merging after Edge Creation)* (described below).

7. If both the distances to the two nearest nodes are within the corresponding similarity thresholds ($||p - p_{s_1}|| \leq thr_{s_1}$ and $||p - p_{s_2}|| \leq thr_{s_2}$):

(a) Increment the age of all edges that are directly connected to node s_1:
$age_{(s_1,n)} = age_{(s_1,n)} + 1 \quad \forall n \in neighbours(s_1)$.

(b) If no edge exists between node s_1 and node s_2, create this edge:
$E = E \cup (s_1, s_2)$.
Execute *(Cluster Merging after Edge Creation)* (described below).

(c) Reset the age of the edge between node s_1 and node s_2:
$age_{(s_1,s_2)} = 0$.

(d) Increase the accumulated error of node s_1 by the distance to the pattern p:
$err_{s_1} = err_{s_1} + ||p_{s_1} - p||$.

(e) Increment the number of signals for node s_1:
$sig_{s_1} = sig_{s_1} + 1$.

(f) Adjust the pattern of node s_1:
$p_{s_1} = p_{s_1} + \epsilon_1 \cdot (p - p_{s_1})$ with $\epsilon_1 = \frac{1}{sig_{s_1}}$.

(g) Adjust the patterns of the neighbours of node s_1:
$p_n = p_n + \epsilon_2 \cdot (p - p_n) \quad \forall n \in neighbours(s_1)$ with $\epsilon_2 = 0.01 \cdot \frac{1}{sig_{s_1}}$.

(h) Remove edges with an age greater than the specified age_{dead}:
$E = E \backslash e \quad \forall e \in E$ with $age_e > age_{dead}$.
Whenever an edge was removed, execute *(Cluster Splitting after Edge Removal)* (described below).

8. If the number of inputs reaches a predefined value λ ($count_p \geq \lambda$), refine and clean up the topology:

   (a) Reset the number of inputs: $count_p = 0$.

   (b) *(Removal of Longest Edge)*

      i. Remove the edge with the maximum length:
      $E = E \backslash e_{max}$ with $e_{max} = \arg\max_{(n_a,n_b)\in E} ||p_a - p_b||$.

      ii. Execute *(Cluster Splitting after Edge Removal)* (described below).

   (c) *(Removal of Minimum-Density Node)*

      i. Remove the node with the minimum number of signals:
      $N = N \backslash n_{min}$ with $n_{min} = \arg\min_{n\in N} sig_n$.

      ii. Remove all edges emanating from node n_{min}:
      $E = E \backslash e \quad \forall e \in E$ with $e = (n_{min}, .)$.
      Whenever an edge was removed, execute *(Cluster Splitting after Edge Removal)* (described below).

      iii. Unassign node n_{min} from its cluster $c_{n_{min}}$: $N_{c_{n_{min}}} = N_{c_{n_{min}}} \backslash n_{min}$.

      iv. Remove the (empty) cluster $c_{n_{min}}$ to which node n_{min} was assigned:
      $C = C \backslash c_{n_{min}}$.

   (d) *(Reduction of Local Error)*
   Reduce the local error by inserting a new node:

      i. Determine the node n_q with the maximum error:
      $n_q = \arg\max_{n\in N} err_n$.

      ii. If node n_q has neighbours:

         A. Determine the neighbour n_f with the maximum error:
         $n_f = \arg\max_{n\in neighbours(n_q)} err_n$.

         B. Create a new node n_r: $N = N \cup n_r$.

         C. Set the pattern for node n_r to $p_r = \frac{1}{2}(p_q + p_f)$.

         D. Set the error for node n_r to $err_r = err_q$.

         E. Set the number of signals for node n_r to $sig_r = sig_q$.

         F. Decrease the error of node n_q: $err_q = 0.5 \cdot err_q$.

         G. Decrease the error of node n_f: $err_f = 0.5 \cdot err_f$.

         H. Create an edge between node n_q and node n_r: $E = E \cup (n_q, n_r)$.

         I. Create an edge between node n_r and node n_f: $E = E \cup (n_r, n_f)$.

         J. Remove the edge between node n_q and node n_f: $E = E \backslash (n_q, n_f)$.

         K. Assign node n_r to the cluster of nodes n_q and n_f: $N_{c_{q,f}} = N_{c_{q,f}} \cup n_r$.

   (e) Prune clusters by removing weakly connected nodes:

      i. Let $\overline{sig}$ denote the average number of signals in the topology:
      $\overline{sig} = \frac{\sum_{n\in N} sig_n}{|N|}$.

ii. Remove nodes with two neighbours based on the number of signals and the c_2 parameter:
   A. $\forall n \in N$: $N = N \backslash n$ if $|neighbours(n)| = 2$ and $sig_n < c_2 \cdot \overline{sig}$.
      Whenever a node n was removed, remove all edges emanating from node n: $E = E \backslash e \quad \forall e \in E$ with $e = (n, .)$.
      Whenever an edge was removed, execute *(Cluster Splitting after Edge Removal)* (described below).
   B. Let $N_{removed}$ denote the set of removed nodes.
   C. Unassign all removed nodes from their clusters:
      $\forall n \in N_{removed} : N_{c_n} = N_{c_n} \backslash n$.
   D. Remove the (empty) clusters to which the removed nodes were assigned: $\forall n \in N_{removed} : C = C \backslash c_n$.

iii. Remove nodes with one neighbour based on the number of signals and the c_1 parameter:
   A. $\forall n \in N$: $N = N \backslash n$ if $|neighbours(n)| = 1$ and $sig_n < c_1 \cdot \overline{sig}$.
      Whenever a node n was removed, remove the edge emanating from node n: $E = E \backslash e$ with $e = (n, .)$.
   B. Let $N_{removed}$ denote the set of removed nodes.
   C. Unassign all removed nodes from their clusters:
      $\forall n \in N_{removed} : N_{c_n} = N_{c_n} \backslash n$.
   D. Remove the (empty) clusters to which the removed nodes were assigned: $\forall n \in N_{removed} : C = C \backslash c_n$.

iv. Remove isolated nodes:
   A. $\forall n \in N$: $N = N \backslash n$ if $|neighbours(n)| = 0$.
   B. Let $N_{removed}$ denote the set of removed nodes.
   C. Unassign all removed nodes from their clusters:
      $\forall n \in N_{removed} : N_{c_n} = N_{c_n} \backslash n$.
   D. Remove the (empty) clusters to which the removed nodes were assigned: $\forall n \in N_{removed} : C = C \backslash c_n$.

(f) *(Joining of Clusters)*
If at least two clusters exist in the topology ($|C| \geq 2$), join the two clusters that are closest to each other, given they are close enough to each other:

i. Determine the two clusters c_a and c_b with the minimum distance between them:
   $c_a, c_b = \arg\min_{c_a, c_b \in C, c_a \neq c_b} ||p_a - p_b||$ with $n_a \in c_a, n_b \in c_b$.

ii. Let d_{min} denote the minimum distance between the clusters c_a and c_b:
   $d_{min} = \min_{c_a, c_b \in C, c_a \neq c_b} ||p_a - p_b||$ with $n_a \in c_a, n_b \in c_b$.

iii. If an absolute join tolerance is used, join the clusters based on the parameter ϕ:

A. If $d_{min} < \sqrt{\phi^2 \cdot dim}$, create an edge between node n_a and node n_b: $E = E \cup (n_a, n_b)$.

iv. If a relative join tolerance is used, join the clusters based on the parameter ψ and the average node distances in clusters c_a and c_b:

A. Let $\overline{d_{c_a}}$ denote the average distance between the nodes of cluster c_a.

B. Let $\overline{d_{c_b}}$ denote the average distance between the nodes of cluster c_b.

C. If $d_{min} < \psi \cdot \overline{d_{c_a}}$ and $d_{min} < \psi \cdot \overline{d_{c_b}}$, create an edge between node n_a and node n_b: $E = E \cup (n_a, n_b)$.
Execute *(Cluster Merging after Edge Creation)* (described below).

D. Repeat *(Joining of Clusters)*.

9. Proceed with the next input (Step 1).

In the following, additional steps of the algorithm are described which are referenced from within the steps above.

- *(Cluster Splitting after Edge Removal)*

1. Let $e = (a, b)$ denote the removed edge.
2. Check if an edge route exists between node a and node b:
Starting from node a, search for node b by recursively following emanating edges. If all connected nodes have been visited and node b was not found, assign the smaller part of the formerly connected nodes to a new cluster:

(a) Let N_a denote the set of nodes that are directly or indirectly connected to node a, including node a. These nodes can be determined by recursively following edges emanating from node a.

(b) Let N_b denote the set of nodes that are directly or indirectly connected to node b, including node b. These nodes can be determined by recursively following edges emanating from node b.

(c) If $|N_a| < |N_b|$, assign the nodes in N_a to a new cluster:

i. Let c_{old} denote the cluster to which the nodes in N_a are assigned.

ii. Unassign all nodes in N_a from cluster c_{old}:
$N_{c_{old}} = N_{c_{old}} \backslash n \quad \forall n \in N_a$.

iii. Create a new cluster c_{new}: $C = C \cup c_{new}$.

iv. Assign all nodes in N_a to cluster c_{new}:
$N_{c_{new}} = N_{c_{new}} \cup n \quad \forall n \in N_a$.

(d) If $|N_a| \geq |N_b|$, assign the nodes in N_b to a new cluster:

i. Let c_{old} denote the cluster to which the nodes in N_b are assigned.

ii. Unassign all nodes in N_b from cluster c_{old}:
$N_{c_{old}} = N_{c_{old}} \backslash n \quad \forall n \in N_b$.

iii. Create a new cluster c_{new}: $C = C \cup c_{new}$.

iv. Assign all nodes in N_b to cluster c_{new}:
$N_{c_{new}} = N_{c_{new}} \cup n \quad \forall n \in N_b$.

- *(Cluster Merging after Edge Creation)*

1. Let $e = (a, b)$ denote the created edge. Let c_a denote the cluster to which node a is assigned. Let c_b denote the cluster to which node b is assigned.
2. If node a and node b belong to different clusters ($N_{c_a} \cap N_{c_b} = \emptyset$):
   (a) If $|N_{c_a}| < |N_{c_b}|$:
      i. Assign all nodes of cluster c_a to cluster c_b:
      $N_{c_b} = N_{c_b} \cup n \quad \forall n \in N_a$.
      ii. Remove cluster c_a: $C = C \backslash c_a$.
   (b) If $|N_{c_a}| \geq |N_{c_b}|$:
      i. Assign all nodes of cluster c_b to cluster c_a:
      $N_{c_a} = N_{c_a} \cup n \quad \forall n \in N_b$.
      ii. Remove cluster c_b: $C = C \backslash c_b$.

3.1.4 Complexity Analysis

The computational complexity of M-SOINN depends mainly on the number of nodes $|N|$, the number of edges $|E|$ and the number of clusters $|C|$ in the topology. Moreover, the computation time of the Euclidean distance depends on the dimensionality *dim* of the input data. Additionally, the computational cost for splitting and merging clusters must be considered. These steps can happen when edges are created or removed. When an edge is created, two clusters may be joined if the connected nodes belong to different clusters. In such a case all nodes of one cluster must be assigned to the other cluster. When an edge is removed, a cluster is split into two if no other edge route exists between the two separated nodes. Thus, from one of the nodes a recursive search must be invoked, which, in the worst case, has to visit all nodes of this cluster. If this cluster holds nearly all the nodes of the topology, all these nodes must be visited. Table 3.1 lists the time complexities of the different processing steps of the M-SOINN algorithm. Many operations have a constant or linear complexity. But some operations have quadratic complexities, which is mostly caused by cluster merging or splitting after the creation or removal of an edge. The steps for connecting a new node, removing old edges, removing the minimum-density node, pruning clusters and joining clusters can become processing bottlenecks. Except for the removal of old edges, corresponding operations can be deactivated to enable a faster operation of the algorithm.

In terms of the topology development, some of the steps lead to network growth or network shrinkage by creating or removing nodes or edges. When removing the

longest edge, the number of edges is reduced by one (and the number of clusters may increase by one). On the other hand, the modification for joining clusters may create in each repetition one additional edge and reduce the number of clusters by one. However, the join process is conditional and depends on the status of the topology. The modification to directly connect new nodes can increase the number of edges by one. In this way, the new node does not form a new cluster and the cluster count remains the same. The cluster pruning step reduces the node count by removing weakly connected nodes (and involved edges) as well as an arbitrary number of isolated nodes (and corresponding one-node clusters). Each removal of an edge can result in a cluster split and, thus, lead to one additional cluster. The removal of the minimum-density node decreases the number of nodes by one and possibly removes a certain number of edges. Its influence on the number of clusters, however, depends on the structure of the topology: if the removed node formed a cluster on its own, the number of clusters decreases by one; if the node connected various other nodes with each other and the node's removal led to a cluster split, then the number of clusters increases. The step for reducing the local error always creates an additional node and increases the edge count by one but never influences the number of clusters.

processing step	time complexity
increment number of inputs	$\mathcal{O}(1)$
number of nodes < 2	$\mathcal{O}(1)$
create new node	$\mathcal{O}(1)$
find two nearest nodes	$\mathcal{O}(\lvert N\rvert \cdot dim)$
update similarity thresholds (fixed)	$\mathcal{O}(1)$
update similarity thresholds (dynamic)	$\mathcal{O}(\lvert N\rvert \cdot dim)$
distance $>$ similarity threshold	$\mathcal{O}(1)$
connect new node (extend cluster check)	$\mathcal{O}(\lvert N\rvert \cdot dim)$
connect new node (average distance check)	$\mathcal{O}(\lvert N\rvert^2 \cdot dim)$
connect new node (edge creation)	$\mathcal{O}(\lvert N\rvert)$
increment age of emanating edges	$\mathcal{O}(1)$
connect nearest nodes	$\mathcal{O}(N)$
update variables of nearest node	$\mathcal{O}(1)$
adjust pattern of nearest node	$\mathcal{O}(dim)$
adjust pattern of neighbours	$\mathcal{O}(\lvert N\rvert \cdot dim)$
remove old edges	$\mathcal{O}(\lvert E\rvert^2 \cdot \lvert N\rvert)$
number of inputs $\geq \lambda$	$\mathcal{O}(1)$
reset number of inputs	$\mathcal{O}(1)$
remove longest edge	$\mathcal{O}(\lvert E\rvert \cdot dim + \lvert E\rvert \cdot \lvert N\rvert)$
remove minimum-density node	$\mathcal{O}(\lvert E\rvert^2 \cdot \lvert N\rvert)$
reduce local error	$\mathcal{O}(\lvert N\rvert + dim)$
prune clusters	$\mathcal{O}(\lvert N\rvert^2 \cdot \lvert E\rvert^2)$
join clusters	$\mathcal{O}(\lvert C\rvert^2 \cdot \lvert N\rvert^2 \cdot dim)$
input processing (all modifications deactivated)	$\mathcal{O}(\lvert N\rvert \cdot dim + \lvert E\rvert^2 \cdot \lvert N\rvert)$
input processing (all modifications activated)	$\mathcal{O}(\lvert N\rvert^2 \cdot dim + \lvert E\rvert^2 \cdot \lvert N\rvert)$
topology refinement (all modifications deactivated)	$\mathcal{O}(\lvert N\rvert^2 \cdot \lvert E\rvert^2)$
topology refinement (all modifications activated)	$\mathcal{O}(\lvert E\rvert \cdot dim + \lvert N\rvert^2 \cdot \lvert E\rvert^2 + \lvert C\rvert^2 \cdot \lvert N\rvert^2 \cdot dim)$

Table 3.1: Time complexity of the M-SOINN processing steps.

3.2 TOSAM: Temporal-Order Sensitive Associative Memory

TOSAM is a memory model that incorporates Hebbian learning to store associations between incoming information. Based on the strengths of the learned associations, TOSAM is able to recall information that has been associated with a given stimulus. The retrieval process is performed by spreading activation to related memory items – an idea that can also be found in other memory models (Anderson, 1983; Lim et al., 2010).

3.2.1 Information Units

The model consists of a network of information units (Figure 3.3). Each information unit stores a *data pattern* (a piece of information) that can be a binary string of an arbitrary length, an integer, a floating point number, or another data type. As a part of ICALA no actual pattern of sensory data is stored but only a unique identifier of the cluster that relates to the given unit. Thus, each information unit stores a symbolic representation of a particular category of inputs. If the model receives an input but no unit with the incoming data pattern (cluster identifier) exists in the network, a new unit is created for this pattern. Additionally, each information unit has two real-valued variables assigned: an *input load* and an *activation level*. The input load represents the perception rate strength of the respective input and drives the learning process. The activation level is the current amount of activation of the respective unit and determines how strongly the corresponding information is recalled.

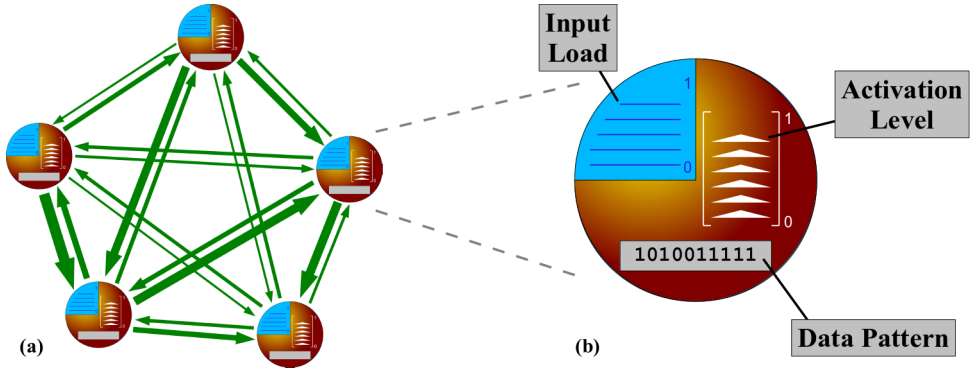


Figure 3.3: Network structure of TOSAM. The network of information units is fully connected by directed associations (a). Each unit stores a data pattern and has an input load and an activation level (b).

Both the input load and the activation level can take values between 0 and 1. When a data pattern is perceived, both the input load l and the activation level a of

the corresponding unit u are raised to their maximum values (Equation 3.1):

$$\begin{aligned} l_u &= 1 \\ a_u &= 1 \end{aligned} \tag{3.1}$$

Otherwise, these values underlie an exponential decay and, thus, decrease to near zero within a few cycles (discrete time steps t) if the respective perception is not present. An exponential function successfully models decay forgetting in short-term memory (Rubin and Wenzel, 1996) and furthermore does not require a parameter of how much time has passed since a unit has last been activated or received an input. The input load l of a unit u decays depending on a constant ζ (Equation 3.2). The decay of a unit u's activation a depends on the constants α and β (Equation 3.3). β is part of a weakening factor that weakens the decay of activation levels close to 1. In this way, a higher activation level is maintained for more cycles.

$$l_u(t+1) = l_u(t) \cdot (1-\zeta), \quad \zeta = 0.2 \tag{3.2}$$

$$a_u(t+1) = a_u(t) \cdot (1 - \alpha \cdot \frac{1}{1+e^{\beta \cdot (a_u(t)-0.5)}}), \quad \alpha = 0.1, \quad \beta = 7 \tag{3.3}$$

3.2.2 Associations/Connections

The network of information units is fully connected. From each unit in the network there are connections to all other units. These connections are directed, which means that between any pair of units there are two connections – one in each direction. Such a connectivity is in favour of the IAH (Section 2.3.3) and offers higher flexibility for representing relationships and modelling sequential activation spreading. A real-valued *connection weight* represents the associative strength between the connected units.

The weight w_{uv} of the connection from unit u to unit v can take values between -1 and 1 and decays slowly towards 0 according to Equation 3.4:

$$w_{uv}(t+1) = w_{uv}(t) \cdot (1-\omega), \quad \omega = 0.000001 \tag{3.4}$$

Although logarithmic and power functions provide a better fit to many empirically established forgetting curves (Wixted and Ebbesen, 1991; Rubin and Wenzel, 1996; Wixted and Ebbesen, 1997), again an exponential function has been chosen here due to its computational properties. The weight change can always be calculated based only on the current weight value.

3.2.3 Learning

Learning in TOSAM is based on the co-occurrence of different input stimuli. An association between two inputs is formed and reinforced when the perceptional strengths of both these inputs are intense. By modifying connection weights between units that have a non-zero input load, the corresponding association strength is altered. An asymmetric Hebbian learning rule allows the model to be sensitive to the exact timings of perceived inputs. For instance, if a certain stimulus is perceived shortly before another one, the association from the former to the latter should be learned stronger than the association in the opposite direction. In principle, this can be achieved by having the two corresponding perceptional strengths contribute in different ways. The stimulus strength where the association originates from should decrease over time, such that a longer gap between the two stimuli results in weaker learning. The stimulus strength at the destination should serve as a trigger for learning – a threshold function ensures that only very high stimulus strength values cause a change in the association weight. The combination of both influences leads to the previously mentioned asymmetric learning effect. Figure 3.4 schematically illustrates the weight update process.

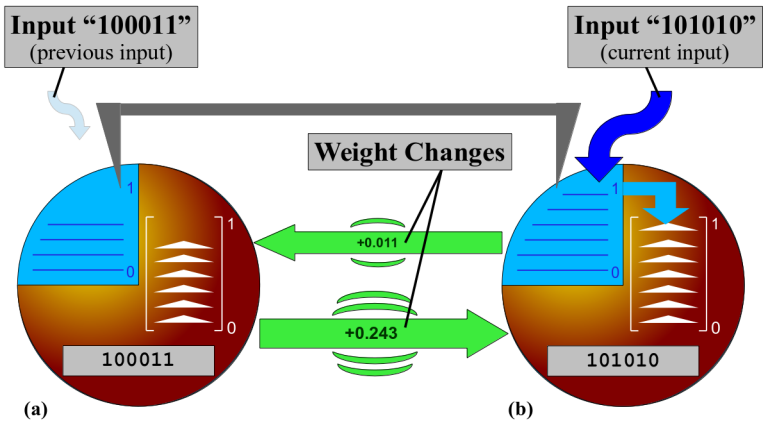


Figure 3.4: Connection weight update in TOSAM. The weight change depends on the input loads of the units that are connected by the corresponding connection. The input ”100011” is no longer present and the input load of the corresponding unit has decreased (a). The unit storing ”101010” has maximum load as this pattern is currently shown as an input (b).

The input loads of the linked units influence the learning process in different ways. A strong change in weight only occurs when the input load of the respective destination unit is reasonably strong. Any remaining (not decayed) input load of the source unit is strengthened, which allows the learning to take effect even if the two stimuli were not perceived at exactly the same time. Hence, if a stimulus A is perceived shortly before a stimulus B, the weight of the connection from A to B will increase strongly whereas the weight of the opposite direction will hardly change. In this way, sequential information and the temporal structure of the stream of inputs

is stored in the network weights. In line with the Rescorla-Wagner theory (Rescorla and Wagner, 1972) the change in weight becomes smaller the closer the weight value is to its maximum.

The change in weight Δw for the connection from unit u to unit v depends on the input loads of both connected units. For unit u, a higher input load l results in a higher contribution *contr* towards the weight change (Equation 3.5). The input load is amplified, controlled by the parameter ξ.

$$contr_u = 1 - (1 - l_u)^{\xi}, \quad \xi = 2 \tag{3.5}$$

Unit v contributes to positive weight changes only when its input load is high (Equation 3.6). The exponentiation with a high value for the exponent ρ suppresses low values and acts like a threshold for learning. If unit v has a very low input load, unlearning of the corresponding association may occur. Also known as the effect of extinction, unlearning occurs when a stimulus is present but a previously associated response is not observed. Here, the intensities of stimulus and response are represented by the input loads of unit u and unit v, respectively. The constant χ determines the strength of the extinction effect. A low value of l_v leads to a negative contribution, which allows the weight value to decrease and eventually let the connection become inhibitory.

$$contr_v = (l_v \cdot (1 + \chi) - \chi)^{\rho}, \quad \chi = 0.8, \quad \rho = 15 \tag{3.6}$$

Both contribution factors are combined multiplicatively and, moreover, changes for stronger weights are weakened (Equation 3.7):

$$\Delta w_{uv} = contr_u \cdot contr_v \cdot (1 - |w_{uv}|) \tag{3.7}$$

Having computed Δw, the weight value is updated according to Equation 3.8 in which the parameter η determines the learning rate.

$$w_{uv}(t + 1) = w_{uv}(t) + \Delta w_{uv}(t) \cdot \eta, \quad \eta = 0.04 \tag{3.8}$$

3.2.4 Recall

The weight value influences how much activation can be spread over the corresponding connection. When the algorithm spreads activation from one unit to another, an amount of activation is removed from the source unit (the unit the connection is originating from) and then transferred to the destination unit (the unit the connection links to). The amount of activation that is currently transferred over the connection is denoted as *signal*. The actual signal value is calculated based on the activation level of the source unit and the weight of the connection to the destination unit.

Moreover, if the destination unit is already highly activated, this unit will attract less activation, resulting in a smaller signal amount being spread to this unit. Finally, low signal values are weakened and high values are strengthened. This serves the purpose of suppressing the recall of weakly associated information and only allows the recall of strongly associated information. If the total amount of activation that a unit needs to spread exceeds its current activation level, the amount of available activation is divided amongst all destination units in proportion to the respective calculated signal values. The process of signal spreading is depicted in Figure 3.5.

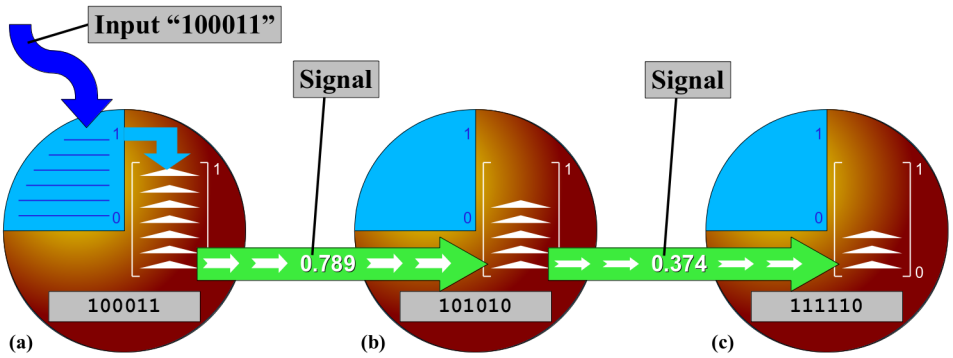


Figure 3.5: Activation spreading in TOSAM. The pattern ”100011” is presented and activates the unit that stores this data pattern (a). Activation is spread from one unit to another (referred to as signal) in accordance to the associative strength in the corresponding direction (b, c). The amount depends on the connection weight and the activation levels of the involved units.

For a connection with weight w the signal s that spreads from a unit u to a unit v is calculated as follows (Equation 3.9):

$$s_{uv} = a_u \cdot attr_v \tag{3.9}$$

where $attr$ is the attraction of the destination unit v (Equation 3.10):

$$attr_v = \begin{cases} |w_{uv}| \cdot (1 - a_v) & \text{if } w_{uv} \geq 0 \\ |w_{uv}| \cdot a_v & \text{if } w_{uv} < 0 \end{cases} \tag{3.10}$$

If the sum of all signals originating from unit u exceeds the current activation level of this unit, all signal values are divided by the sum of attractions of all outgoing connections (Equation 3.11):

$$s_{uv} = \frac{a_u \cdot attr_v}{\sum_{x \in destinations_u} attr_x} \tag{3.11}$$

Furthermore, the signal is processed in a sigmoidal way, resulting in the processed signal s' (Equation 3.12):

$$s'_{uv} = (1 - s_{uv})^4 - 2 \cdot (1 - s_{uv})^2 + 1 \tag{3.12}$$

This is the amount of activation that is spread over the connection from unit u to unit v. If the weight w_{uv} is negative, the amount needs to be subtracted in the destination unit; or simply $-s'_{uv}$ is spread over the connection.

3.2.5 Clean-Up Process

As a new unit is created whenever new information is received, the total number of units (and connections) in the network can become infinitely large. In order to avoid such a situation, insignificant units are removed from the network. A unit can be considered insignificant when all the weights of connections that link with this unit are extremely small. This unit does not significantly contribute to any spreading process in the network. If, additionally, the respective input load is considerably small, the unit is not significantly involved in any learning process and, thus, can be removed without having any noticeable effect on the whole network. A unit u and corresponding connections $connections_u$ are removed if $l_u < 0.01$ and $\forall\, connection \in connections_u : |w_{connection}| < 0.01$.

3.2.6 Processing Cycle

Both learning and recall are ongoing processes that are repetitively executed by the model in short cycles. The interval between cycles is intended to be short enough to enable a near real-time operation of the model. There should not be any noticeable delay when processing perceived stimuli, which requires a cycle time within the range of milliseconds. In each cycle the operations described above are executed in the order listed below. For every step also the time complexity is given with $|U|$ denoting the number of units and $|A|$ the number of associations/connections. In a fully-connected network the number of associations is $|A| = 2 \cdot \binom{|U|}{2} = 2 \cdot \frac{(|U|-1)\cdot|U|}{2} = (|U|-1)\cdot|U|$.

1. Each unit leaks its input load (Equation 3.2). Looping over all units has a time complexity of $\mathcal{O}(|U|)$.

2. Each unit decays its activation (Equation 3.3). The time complexity of this step is $\mathcal{O}(|U|)$.

3. Each connection decays its weight (Equation 3.4). This step has a time complexity of $\mathcal{O}(|A|)$.

4. All perceived inputs are read. If, for any input, no unit exists in the network, a new unit with the respective data pattern is created. The time complexity for the creation of one unit and corresponding connections is $\mathcal{O}(|U|)$. Then both input load and activation level are raised (Equation 3.1) in all units corresponding to the current inputs.

5. Activation is spread over all connections. For each connection, first the computed signal (Equation 3.12) is removed from the source unit's activation and transferred onto the respective connection. Then the signals are removed from all connections and added to the activation values of respective destination units. This process has a time complexity of $\mathcal{O}(|A|)$.

6. Learning takes place by updating the weight values of all connections (Equation 3.8). This step has a time complexity of $\mathcal{O}(|A|)$.

7. The network is cleaned up by removing insignificant units. Looping over all units and connected associations has a time complexity of $\mathcal{O}(|U| \cdot |A|)$.

3.3 Synchronisation of M-SOINN and TOSAM

TOSAM needs to reflect with its units the cluster structure present in the M-SOINN topology. Only clusters that contain a certain minimum number of nodes are represented in TOSAM and can activate the corresponding unit. This parameter (*cluster activation threshold*) for activating a unit needs to be predefined for each M-SOINN module. After each processing cycle, an M-SOINN module notifies TOSAM about the current activated cluster (if that cluster contains enough nodes), which is done by passing a unique identifier. Thus, M-SOINN has to maintain an identifier for each cluster that must remain consistent even if the topological structure changes. Furthermore, TOSAM needs to update learned associative strengths accordingly. This is achieved by applying the following rules simultaneously.

1. Whenever a node is created, also a cluster with a new identifier is created and the new node is assigned to this cluster. Thus, an isolated node already forms a cluster. If the respective module is configured such that already a one-node cluster creates an input for TOSAM, a new unit and connections need to be created in TOSAM. Creating connections to all other units has a time complexity of $\mathcal{O}(|U|)$.

2. Whenever a node is removed, the corresponding cluster is checked for its number of nodes. If empty, this cluster is removed as well as the corresponding unit in TOSAM. The removal of all connections with this unit in TOSAM has a time complexity of $\mathcal{O}(|U|)$, if each unit maintains a list of only the connections involving the unit itself.

3. Whenever an edge is created, two clusters may merge into one. This happens if the connected nodes belong to different clusters. The merging is done by keeping the cluster that contains more nodes and assigning to it all nodes of the other cluster. The cluster with the smaller number of nodes is then removed. In TOSAM the corresponding units and connections need to be merged as

well. The input load, activation level and association strengths are updated by taking, in each case, the maximum of the former two values. The time complexity of this step is $\mathcal{O}(|U|^2)$ as, for each connection with the unit to be kept ($\mathcal{O}(|U|)$), TOSAM needs to loop over all connections with the unit to be deleted to identify the matching connection ($\mathcal{O}(|U|)$).

4. Whenever an edge is removed, a cluster may be split into two. This happens if no other route exists between the nodes that were connected by the removed edge. In this case, M-SOINN maintains the cluster identifier for the part that contains more nodes and assigns an entirely new identifier to the other part. For the new cluster TOSAM creates a new unit as well as connections. For its input load and activation level, the new unit adopts the exact values of the already existent unit. Similarly, weight values for created connections are copied from the corresponding existing connections. The creation of a new unit and corresponding connections has a time complexity of $\mathcal{O}(|U|)$.

3.4 Summary

In this chapter the structural and functional details of ICALA and of its main components M-SOINN and TOSAM were fully described. The complete M-SOINN algorithm was given and all equations of the TOSAM model were listed. Also explained were the steps for synchronising these two components. The following chapters separately provide evaluations of M-SOINN and TOSAM before the functionality of the entire architecture is validated.

Chapter 4

Evaluations on Incremental Clustering

The M-SOINN algorithm as described in Section 3.1 improved the original SOINN algorithm in several ways. Evaluations were performed to test the impact of the introduced modifications. This chapter explains the evaluation procedures and presents and discusses corresponding results. The chapter also provides a brief comparison with previous results of SOINN and E-SOINN.

The introduced modifications were evaluated on two datasets, namely the *MNIST database of handwritten digits*[1] and the *AT&T database of faces*[2]. The former contains 70000 greyscale images of handwritten digits with different variations in writing. The latter is made up of portrait images of 40 different people and for each person the dataset contains ten images in which the head rotation, facial expression, etc. vary.

For each of the two datasets, three scenarios were tested: *Random*, *Ordered-Random* and *Permuted-Random*. The scenarios differed in the way the inputs were presented. In the *Random* scenario the next input was chosen completely randomly from all available inputs. In the *Ordered-Random* scenario the inputs of one category were presented successively but within each category the input order was random. The order of categories, however, was always the same. The scenario *Permuted-Random* randomised the category order for every new run and, within each category, the next input was chosen randomly.

Each scenario was divided into three phases. In phase 1 a baseline for the clustering performance was determined by testing the M-SOINN algorithm without using any of the introduced modifications. Instead, the evaluations in this phase were done for various parameter settings. All combinations of the following values were tested:

[1] `http://yann.lecun.com/exdb/mnist/`

[2] `http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html`

- $age_{dead} \in \{10, 100, 1000\}$
- $\lambda \in \{10, 25, 50, 100\}$
- $c_2 \in \{0.01, 0.02, 0.05, 0.1\}$
- $c_1 \in \{0.1, 0.2, 0.5, 1.0\}$

These are $3 \cdot 4 \cdot 4 \cdot 4 = 192$ combinations in total. Only the parameter values that led to the best results (with the lowest overall score) were chosen for phase 2 where the different modifications were tested. As each modification can be enabled or disabled independently of the others, all possible combinations could be tested, including the algorithm with all modifications disabled. Thus, evaluated were all combinations of the introduced modifications ($2^5 = 32$ combinations):

- Connection of New Node *(NodeNewConnection, NNC)*
- Removal of Longest Edge *(EdgeMaxRemoval, EMR)*
- Removal of Minimum-Density Node *(NodeMinRemoval, NMR)*
- Reduction of Local Error *(ReduceErrorInsertion, REI)*
- Joining of Clusters *(ClusterJoining, CJ)*

Finally, in phase 3 an actual topology was examined in detail, obtained when using the best parameter settings and the modification combination that led to good results in phase 2.

As the testing included randomness, for each setting several runs were performed. In phase 1, these were 100 runs for each parameter configuration, and in phase 2 another 100 runs for each modification combination. Phase 3 consisted of only 1 run as only one topology was required in order to provide an example of a possible clustering result. In every single run, M-SOINN received a total of 1000 inputs.

For phases 1 and 2, the minimum, the maximum and the mean average (over 100 runs) are reported for both the number of clusters and the number of represented digits or people, respectively. The number of clusters can be determined straightaway by the algorithm. The number of different digits or people (i.e. different categories) was determined based on a majority vote for each cluster by counting how often each category was represented in each cluster. The category with the highest count was chosen as the category of the entire cluster. This required to store a category label together with every input. Normally M-SOINN stores only a pattern for each node without any label information. Moreover, the position of a node (i.e. its pattern) can change during the course of learning. Thus, a node cannot be linked to its original input pattern or the category from which this pattern originated. In order to determine the category of a node, a corresponding label needed to be stored. When

presenting an input, at the same time a category label was passed to the algorithm. Thus, every node in the topology also had a category label assigned. These labels were only included for the purpose of evaluation but are not required for learning, which still happened in an unsupervised fashion.

For all evaluations no noise was added to the inputs, the similarity threshold was dynamic and the cluster join threshold was relative with $\psi = 1.0$.

4.1 Score Metrics

For scoring the overall clustering result, two score metrics were employed: one for the number of generated clusters and one for the number of represented categories. Both metrics calculate the corresponding score based on the minimum, maximum and average values of the cluster counts or the category counts, respectively. In the following, a general notation is given. For computing the cluster score the corresponding cluster counts should be inserted. Similarly, for the category score the corresponding category counts must be used.

Each metric considers the deviation between the actual number of clusters/categories and the desired number of clusters/categories (Equations 4.1 to 4.4). The desired number for both clusters and categories was set to the number of actually presented categories. For instance, if 10 different categories are presented, the desired topology should contain 10 clusters that represent all these categories. The value 1 is added to all counts in order to avoid a result of 0, which would impact further multiplications and lead to an overall score of 0.

$$deviation_{minimum} = |count_{minimum} - count_{desired}| + 1 \tag{4.1}$$

$$deviation_{average} = |count_{average} - count_{desired}| + 1 \tag{4.2}$$

$$deviation_{maximum} = |count_{maximum} - count_{desired}| + 1 \tag{4.3}$$

$$deviation = deviation_{minimum} \cdot deviation_{average} \cdot deviation_{maximum} \tag{4.4}$$

Moreover, the metrics penalise cluster/category counts below the corresponding desired number (Equations 4.5 to 4.8). If the topology contains less clusters than the number of presented categories, there is no possibility of correctly representing all categories.

$$penalty_{minimum} = \begin{cases} 10 & \text{if } count_{minimum} < count_{desired} \\ 1 & \text{otherwise} \end{cases} \tag{4.5}$$

$$penalty_{average} = \begin{cases} 100 & \text{if } count_{average} < count_{desired} \\ 1 & \text{otherwise} \end{cases} \tag{4.6}$$

$$penalty_{maximum} = \begin{cases} 1000 & \text{if } count_{maximum} < count_{desired} \\ 1 & \text{otherwise} \end{cases} \tag{4.7}$$

$$penalty = penalty_{minimum} \cdot penalty_{average} \cdot penalty_{maximum} \tag{4.8}$$

Furthermore, both metrics favour low variance over numerous runs (Equation 4.9) as it is desirable to obtain similar topologies independent of the exact order of inputs.

$$variance = count_{maximum} - count_{minimum} + 1 \tag{4.9}$$

For calculating the cluster/category score, the results of the previously described equations are multiplied together (Equation 4.10).

$$score = deviation \cdot penalty \cdot variance \tag{4.10}$$

Finally, both cluster score and category score are combined to an overall score by multiplying both score values with each other (Equation 4.11).

$$overall\ score = cluster\ score \cdot category\ score \tag{4.11}$$

The lower the score value, the better is the result according to these metrics.

4.2 MNIST Database of Handwritten Digits

The MNIST database of handwritten digits contains greyscale images of the size 28 by 28 pixels. Thus, M-SOINN received input vectors of the dimensionality 784. For the evaluations performed here only the training set of the MNIST database was considered, which consists of 60000 images. As there are 10 different digits in the training set, the desired cluster count and the desired category count were set to 10. In this dataset, each digit represents a category. Thus, the category score is called digit score in this section.

4.2.1 Scenario 1: Random

In this scenario an image was chosen randomly as the next input for M-SOINN. This means that the algorithm received the different digits in a random order and any subsequent inputs were likely to be different digits.

To establish a baseline, the clustering performance was evaluated without using any of the introduced modifications. Considering only the ten configurations with

the lowest overall score, for each of these configurations, the minimum number of clusters was either 1 or 2, the maximum number of clusters ranged from 14 to 19. Likewise, the minimum number of different digits represented was either 1 or 2, but for each parameter configuration there was at least one run where 10 different digits were represented in the topology. Although sometimes the topology contained the correct number of 10 digits, none of the parameter settings could guarantee satisfactory results. All of these settings could produce a topology with only 1 or 2 clusters, which prevents the possibility of representing 10 different digits. Hence, the M-SOINN algorithm without any modifications fails to generate good clustering results by falling below the minimum required number of clusters. The lowest-scoring results were obtained for a low $age_{dead} = 10$ and a high $\lambda = 100$, with $c_1 = 0.02$ and $c_2 = 0.1$.

NNC	EMR	NMR	REI	CJ	cluster count			digit count			cluster score	digit score	overall score
					min	max	avg	min	max	avg			
1	**0**	**0**	**1**	**0**	22	50	35.3	8	10	9.6	407137.4	12330.0	5020003895.4
1	0	0	1	1	13	59	36.0	7	10	9.6	253518.0	22400.0	5678803200.0
1	0	0	0	0	14	56	31.4	7	10	9.4	226756.2	26400.0	5986363680.0
1	1	0	1	0	20	56	36.7	8	10	9.7	529490.7	11340.0	6004424764.8
1	1	1	0	0	18	48	31.9	7	10	9.5	249174.9	24800.0	6179537520.0
1	1	0	1	1	16	52	36.0	7	10	9.7	301033.1	20960.0	6309653985.6
1	0	1	1	1	18	61	36.4	8	10	9.7	564426.7	11430.0	6451397409.6
1	0	1	1	0	22	56	36.7	8	10	9.7	593219.9	11700.0	6940672830.0
1	1	0	0	0	17	52	30.9	7	10	9.4	271333.4	25600.0	6946136064.0
1	0	1	0	1	17	52	31.6	7	10	9.4	279878.4	26080.0	7299228672.0
1	1	1	0	1	18	52	31.3	7	10	9.4	302189.0	24960.0	7542636192.0
1	1	1	1	0	20	62	36.7	8	10	9.7	694411.3	11700.0	8124612210.0
1	0	0	0	1	18	46	30.3	6	10	9.2	205887.2	44250.0	9110510370.0
1	1	0	0	1	15	54	29.9	6	10	9.3	226152.0	43000.0	9724536000.0
1	1	1	1	1	20	60	35.1	7	10	9.6	599406.1	22720.0	13618505683.2
1	0	1	0	0	17	55	31.5	6	10	9.3	323207.0	42750.0	13817100960.0
0	0	0	1	0	1	18	8.4	1	10	5.9	4195800.0	506000.0	2123074800000.0
0	1	1	0	1	1	14	5.7	1	10	4.4	3696000.0	656000.0	2424576000000.0
0	1	0	1	0	2	16	8.4	2	9	5.9	2466450.0	738720000.0	1822015944000000.0
0	0	0	0	1	1	13	6.0	1	9	4.4	2626000.0	1182600000.0	3105507600000000.0
0	0	1	1	0	1	15	7.5	1	9	5.4	3132000.0	1008000000.0	3157056000000000.0
0	1	1	1	0	2	16	6.8	2	9	5.2	3969000.0	842400000.0	3343485600000000.0
0	0	1	0	0	1	14	6.6	1	9	4.8	3066000.0	1108800000.0	3399580800000000.0
0	1	0	0	0	1	16	7.5	1	9	5.6	3886400.0	973800000.0	3784576320000000.0
0	***0***	***0***	***0***	***0***	1	15	7.0	1	9	5.1	3627000.0	1054800000.0	3825759600000000.0
0	1	0	1	1	1	15	7.0	1	9	4.9	3645000.0	1092600000.0	3982527000000000.0
0	1	1	1	1	1	14	6.0	1	9	4.5	3486000.0	1162800000.0	4053520800000000.0
0	1	1	0	0	1	15	6.4	1	9	5.0	4167000.0	1083600000.0	4515361200000000.0
0	1	0	0	1	1	15	6.1	1	9	4.6	4410000.0	1150200000.0	5072382000000000.0
0	0	1	0	1	1	16	5.6	1	9	4.3	6003200.0	1200600000.0	7207441920000000.0
0	0	0	1	1	1	18	6.8	1	9	4.9	6836400.0	1103400000.0	7543283760000000.0
0	0	1	1	1	1	17	6.1	1	9	4.6	6609600.0	1161000000.0	7673745600000000.0

Table 4.1: Results of the modifications evaluation for the scenario *Random* with the *MNIST digits* dataset. Listed are all results, sorted by overall score in ascending order. The baseline combination (without any modifications) is highlighted in bold italics. The combination chosen for phase 3 is underlined and highlighted in bold.

In phase 2 all combinations of modifications were evaluated with these parameters. Table 4.1 shows the results for all combinations, sorted by overall score in ascending order. For the top entries, the topology never contained less than 10 clusters. This is the case for any combination where the NNC modification was used. For these combinations, the minimum number of clusters ranged from 13 to 22. Consequently, the number of represented digits was higher compared to the baseline results and,

whenever the NNC modification was enabled, the average number of digits was between 9 and 10. Using only the NNC modification but no other modification, the third-best overall score amongst all combinations was achieved. Better results with lower scores were obtained when additionally using the REI modification. In these cases, however, the minimum number of digits was below 10, which means that still the desired clustering result with 10 different digits cannot be guaranteed. Furthermore, with the NNC modification enabled, the maximum cluster count was considerably high with values from 46 to 61. This is roughly five times as many as the minimally required number of 10 clusters. Thus, more digits are represented in the topology at the expense of a higher number of clusters.

In the previous phase the best results (with lowest overall score) were achieved when using the modifications NNC and REI. Figure 4.1 shows a topology obtained when using these modifications. The other parameters were left unchanged. The topology contained 45 clusters and each of the 10 digits was represented by at least one cluster. Most clusters contained only 2 instances and represented only a certain variation of the corresponding digit. Four clusters contained 4 to 6 instances each, which is still insufficient for representing a whole digit category. The topology contained only three bigger clusters with 11, 21 and 30 instances. The represented digits are "5", "6" and "9", respectively. Indeed, these clusters form better generalisations for the contained digits and can account for a higher variability in writing the corresponding digit.

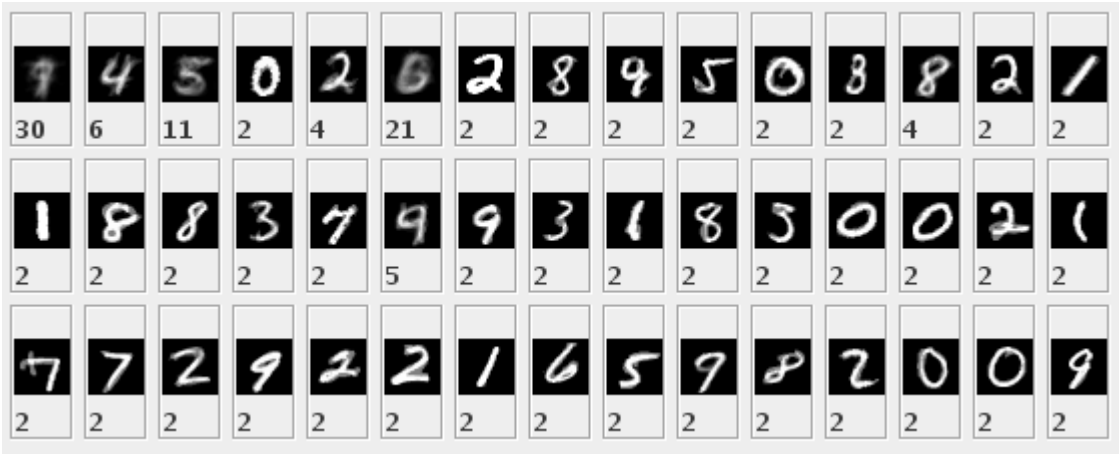


Figure 4.1: Sample topology for the scenario *Random* with the *MNIST digits* dataset. Each rectangle represents a different cluster. The number in each rectangle indicates the number of nodes contained in this cluster and the image visualizes the cluster mean.

4.2.2 Scenario 2: Ordered-Random

In the *Ordered-Random* scenario the digits were presented in their numerical order. First all inputs for the digit "0" were presented, then for the digit "1", and so on. For

each digit, however, the actual order of the corresponding images was randomised. This means that the overall order of images in different runs was different. The total number of inputs in each run was 1000, which corresponds to 100 inputs per digit.

First, evaluations were done without using any modifications. The best (lowest-scoring) result was achieved with the parameters $age_{dead} = 100$, $\lambda = 50$, $c_2 = 0.02$, $c_1 = 0.2$. For this setting, every run led to a digit count between 7 and 10, and the average number of represented digits was 9.65, which is rather close to the desired number of 10. With no modifications the algorithm already produced acceptable clustering results for this scenario.

NNC	**EMR**	**NMR**	**REI**	**CJ**	**cluster count**			**digit count**			**cluster score**	**digit score**	**overall score**
					min	**max**	**avg**	**min**	**max**	**avg**			
1	1	0	0	1	55	104	79.2	10	10	10.0	15343070.0	1.0	15343070.0
1	1	0	1	1	59	106	85.0	10	10	10.0	17699784.0	1.0	17699784.0
1	1	0	1	0	60	107	83.9	10	10	10.0	17973607.7	1.0	17973607.7
1	0	0	1	1	60	114	87.0	10	10	10.0	22975895.3	1.0	22975895.3
0	**1**	**0**	**1**	**0**	20	36	28.9	9	10	9.9	100273.1	4320.0	433179964.8
0	1	0	0	0	16	41	28.3	9	10	9.8	112286.7	4720.0	529993318.4
0	1	1	1	1	14	30	21.0	7	10	9.4	21348.6	26240.0	560187264.0
0	1	0	0	1	12	40	27.2	8	10	9.7	49085.4	11970.0	587552238.0
0	1	1	0	0	14	34	22.0	7	10	9.3	34203.8	26880.0	919396800.0
0	0	1	1	0	18	37	25.9	8	10	9.7	85176.0	11610.0	988893360.0
0	1	1	1	0	14	37	22.4	7	10	9.4	45124.8	25120.0	1133534976.0
0	1	0	1	1	13	41	27.1	7	10	9.8	67113.0	19840.0	1331521126.4
0	0	1	1	1	13	40	23.3	6	10	9.5	49788.5	38250.0	1904409360.0
0	0	0	1	0	20	43	31.2	8	10	9.9	199087.7	10260.0	2042639596.8
0	***0***	***0***	***0***	***0***	16	39	28.8	7	10	9.6	99590.4	21920.0	2183021568.0
0	0	0	1	1	13	42	29.4	6	10	9.8	80704.8	31000.0	2501848800.0
0	1	1	0	1	8	28	19.8	6	10	8.9	129156.3	52000.0	6716127600.0
0	0	1	0	1	8	33	21.0	5	10	9.0	224078.4	72360.0	16214313024.0
0	0	0	0	1	9	43	26.5	5	10	9.4	416500.0	56880.0	23690520000.0
0	0	1	0	0	9	40	23.4	4	10	9.2	285894.4	90650.0	25916327360.0
1	1	1	0	1	44	94	70.3	9	10	9.9	9299225.3	4360.0	40544622090.0
1	1	1	1	1	54	94	74.3	9	10	10.0	10240672.5	4160.0	42601197600.0
1	1	1	1	0	54	100	75.7	9	10	10.0	12827792.3	4040.0	51824280690.0
1	0	1	0	0	53	99	75.8	9	10	9.9	12427232.4	4400.0	54679822560.0
1	0	1	1	1	58	104	81.6	9	10	9.9	15883791.0	4360.0	69253328760.0
1	1	0	0	0	56	110	79.1	9	10	10.0	18291615.1	4040.0	73898125004.0
1	0	0	0	0	49	109	83.6	9	10	10.0	18192640.0	4080.0	74225971200.0
1	0	0	0	1	55	122	81.2	9	10	10.0	25530704.7	4040.0	103144047068.8
1	1	1	0	0	48	98	70.7	8	10	9.9	10918655.3	9720.0	106129329321.6
1	0	0	1	0	63	124	91.5	9	10	10.0	31771850.4	4040.0	128358275616.0
1	0	1	0	1	45	102	75.2	8	10	9.9	12854980.8	10260.0	131892103008.0
1	0	1	1	0	61	102	80.3	8	10	10.0	14490010.1	9360.0	135626494348.8

Table 4.2: Results of the modifications evaluation for the scenario *Ordered-Random* with the *MNIST digits* dataset. Listed are all results, sorted by overall score in ascending order. The baseline combination (without any modifications) is highlighted in bold italics. The combination chosen for phase 3 is underlined and highlighted in bold.

Then with these parameter settings all modification combinations were evaluated. Corresponding results are listed in Table 4.2. For the combinations that led to the four best results (i.e. with the lowest scores), the M-SOINN algorithm always generated topologies that, in every run, contained clusters for all 10 digits. This is ideal because every digit is represented in the topology. But also for these four combinations, the cluster count in each of the resulting topologies was very high with 55 or more clusters. In certain runs, the topology contained more than 100

clusters. Those discussed combinations always included the NNC modification and did not use the NMR modification. The two combinations that produced the next best results (ranked 5th and 6th according to ascending overall score) do not include the modifications NNC, NMR and CJ but include the EMR modification. In these cases the resulting topologies contained a lower number of clusters: between 20 and 36 clusters when the REI modification was used and without the REI modification between 16 and 41 clusters. The average number of clusters for both cases was roughly 28. For both these combination settings, the number of represented digits was at least 9. Thus, in this scenario a good topology can be achieved when using the EMR modification but not enabling the NNC, NMR and CJ modifications.

Although the lowest overall score was obtained with the modifications NNC, EMR and CJ, a topology generated by using the modifications EMR and REI was examined in phase 3. The latter combination led to lower numbers of clusters and almost the same number of represented digits compared to the former combination (the minimum digit count was 9 compared to 10). In the executed run the algorithm produced a topology with 29 clusters and 10 digits, which is shown in Figure 4.2. The majority of clusters consisted of only 2 instances. This low number of instances is insufficient for adequately representing a digit. In these cases, the inherent variability in writing (e.g. rotation or shear) resulted in a digit to be distributed over more clusters. Hence, the topology contained several clusters per digit. There were 12 clusters that contained between 3 and 8 instances, which still accounts for only a small amount of the possible variability in writing the corresponding digit. Only one cluster contained 16 instances, which comes closer to including enough variability for representing an entire digit category. The input data consisted of 100 inputs per digit. Thus, there could be up to 100 different variations per digit.

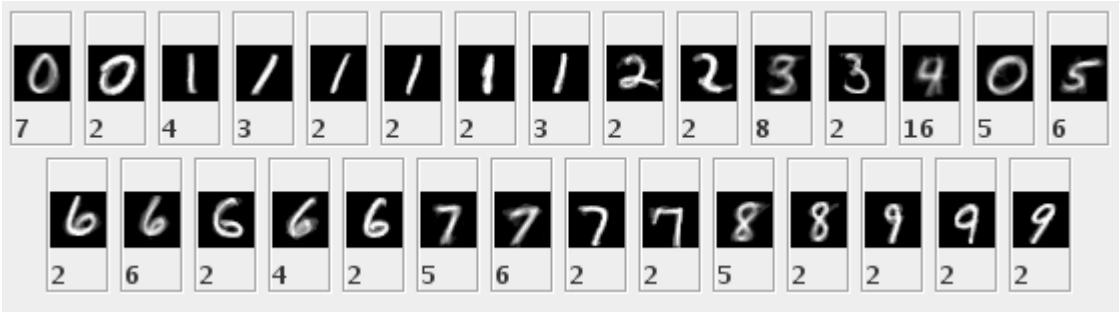


Figure 4.2: Sample topology for the scenario *Ordered-Random* with the *MNIST digits* dataset. Each rectangle represents a different cluster. The number in each rectangle indicates the number of nodes contained in this cluster and the image visualizes the cluster mean.

4.2.3 Scenario 3: Permuted-Random

In the scenario *Permuted-Random* the order in which the digit categories were presented was randomised in each run. The different categories were presented one

after the other. For each digit category the corresponding digits were presented in a random order. For instance, at first, different images of the digit "7" were presented to M-SOINN, then different images of the digit "4", and so on. Per run the algorithm received 1000 inputs, hence, 100 inputs of each of the 10 digits.

In phase 1 different parameter settings without using the modifications were evaluated. Considering only the parameter settings that achieved the ten lowest overall scores, the algorithm always generated topologies with at least 10 clusters. For almost all of these parameter settings, there were some runs during which the topology contained more than 40 clusters. For the lowest-scoring result with $age_{dead} = 1000$, $\lambda = 50$, $c_2 = 0.02$, $c_1 = 0.2$, the digit count ranged from 6 to 10 and the average number of represented digits was 9.3. Despite a rather high number of clusters produced in some runs, the algorithm without any modifications did not achieve to always represent all 10 digits in the topology.

NNC	EMR	NMR	REI	CJ	cluster count			digit count			cluster score	digit score	overall score
					min	max	avg	min	max	avg			
1	**1**	**0**	**1**	**1**	60	109	85.9	10	10	10.0	19619700.0	1.0	19619700.0
1	0	0	1	0	69	118	91.1	10	10	10.0	26859780.0	1.0	26859780.0
0	1	1	0	1	10	30	20.4	6	10	8.8	5005.4	56250.0	281550937.5
0	1	1	1	1	12	30	20.6	6	10	9.0	13933.1	49000.0	682720920.0
0	1	1	1	0	15	32	22.5	7	10	9.5	33583.7	24640.0	827501875.2
0	1	0	1	0	19	37	28.4	8	10	9.8	102942.0	10620.0	1093244040.0
0	0	1	1	1	12	35	23.5	6	10	9.2	27087.8	45500.0	1232496720.0
0	1	1	0	0	14	32	22.2	6	10	9.3	28863.9	43750.0	1262793437.5
0	0	1	0	0	14	35	24.3	7	10	9.1	43872.4	29920.0	1312662208.0
0	0	1	1	0	16	36	25.7	7	10	9.5	66361.7	24000.0	1592680320.0
0	***0***	***0***	***0***	***0***	11	40	27.4	5	10	9.5	34131.0	55080.0	1879935480.0
0	0	0	1	1	11	41	26.6	5	10	9.3	34819.2	60840.0	2118400128.0
0	0	0	1	0	20	43	31.6	8	10	9.8	203126.9	10800.0	2193770304.0
0	1	0	1	1	15	38	27.5	6	10	9.7	77047.2	32250.0	2484772200.0
0	1	0	0	0	18	40	28.4	7	10	9.7	124425.6	20160.0	2508420700.8
0	1	0	0	1	9	38	26.2	7	10	9.4	299454.0	24960.0	7474371840.0
0	0	0	0	1	6	42	25.3	6	10	9.2	992673.0	46000.0	45662958000.0
1	1	0	0	0	52	103	80.9	9	10	10.0	15120637.0	4040.0	61087373318.4
1	1	0	1	0	66	105	87.3	9	10	10.0	17136115.2	4040.0	69229905408.0
1	0	0	0	0	48	112	81.3	9	10	10.0	18872669.4	4120.0	77755397928.0
1	0	0	0	1	46	114	80.8	9	10	10.0	19239025.1	4160.0	80034344208.0
1	0	1	1	1	58	115	83.2	9	10	9.8	22358923.4	4840.0	108217189449.6
1	0	0	1	1	65	124	88.6	9	10	10.0	30769032.0	4040.0	124306889280.0
1	1	1	0	0	50	96	72.3	8	10	9.7	10608828.7	11880.0	126032885193.6
1	1	0	0	1	54	101	79.4	8	10	10.0	13995849.6	9270.0	129741525792.0
1	1	1	0	1	40	99	70.5	8	10	9.5	10293426.0	13230.0	136182025980.0
0	0	1	0	1	3	34	21.4	3	10	8.3	794880.0	173440.0	137863987200.0
1	1	1	1	1	49	101	77.0	8	10	9.8	13270521.6	11070.0	146904674112.0
1	1	1	1	0	54	102	78.1	8	10	9.8	14169991.5	10620.0	150485309730.0
1	0	1	1	0	66	105	83.5	8	10	9.8	16308748.8	10800.0	176134487040.0
1	0	1	0	1	39	103	73.7	8	10	9.3	11852178.0	14940.0	177071539320.0
1	0	1	0	0	45	103	75.6	7	10	9.5	13297089.6	23520.0	312747547392.0

Table 4.3: Results of the modifications evaluation for the scenario *Permuted-Random* with the *MNIST digits* dataset. Listed are all results, sorted by overall score in ascending order. The baseline combination (without any modifications) is highlighted in bold italics. The combination chosen for phase 3 is underlined and highlighted in bold.

Phase 2 evaluated all modification combinations for the mentioned parameter settings. Table 4.3 lists the results for all modification combinations, sorted by overall score in ascending order. For the two combinations with the lowest overall scores,

the generated topology always contained 10 different digits. But for producing these results, the algorithm also generated a very high number of clusters. In both these cases, the minimum cluster count is at least 60 with more than 85 clusters generated on average. The modifications included in these combinations are NNC and REI but NMR was not used in either case. All of the 15 next best results (ranked 3rd to 17th according to ascending overall score) do not employ the NNC modification and the minimum and maximum cluster counts are much lower compared to the two results with the lowest scores. This suggests that mainly the NNC modification causes the algorithm to generate more clusters. But none of the topologies that were generated without using the NNC modification contained the desired number of 10 different digits in every run; the best minimum digit count achieved without the NNC modification was 8. This happened when only using the modifications EMR and REI, but no other modifications.

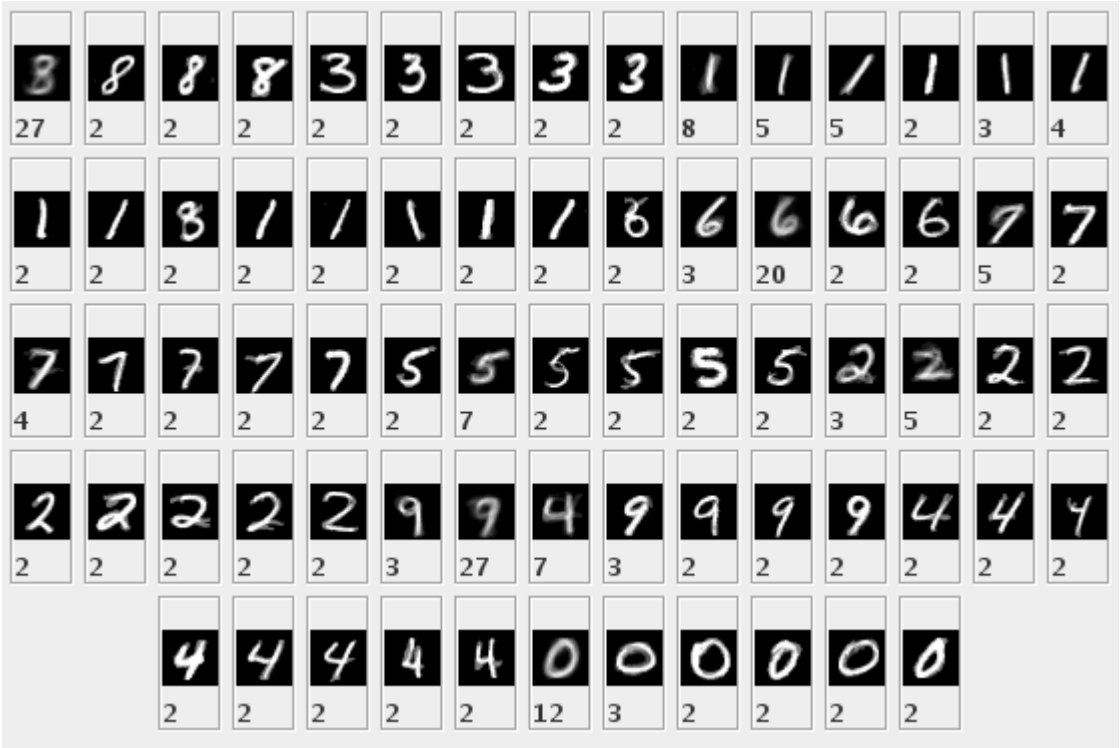


Figure 4.3: Sample topology for the scenario *Permuted-Random* with the *MNIST digits* dataset. Each rectangle represents a different cluster. The number in each rectangle indicates the number of nodes contained in this cluster and the image visualizes the cluster mean.

Examined in phase 3 was a topology in which all 10 different digits could be represented. According to the previous phase, this could only be guaranteed for the combinations that produced the two lowest overall scores, as for these combinations the minimum digit count was 10. The combination with the lowest overall score was chosen to execute one run. The employed modifications were NNC, EMR, REI and CJ. The resulting topology contained 71 clusters which represented all 10 digits (Figure 4.3). The topology contained four bigger clusters with 12, 20, 27

and 27 instances. Two of these clusters clearly represented the digits "0" and "6", respectively. The other bigger clusters with 27 instances each contained instances of two different digits. In one case these were the digits "8" and "3", in the other case the digits were "9" and "7". Here the algorithm falsely formed clusters that incorrectly represented more than one digit. These clusters were not split into separate clusters, although the EMR modification was used. 15 clusters contained between 3 and 8 instances, which still do not capture lots of the variability present in the writing of the corresponding digits. The majority of clusters was made up of only 2 instances. Each of these clusters alone stored only a certain way of writing the corresponding digit. But due to the fact that each digit was represented by several clusters, still various ways of writing the digits were represented in the topology.

4.3 AT&T Database of Faces

The AT&T database of faces contains greyscale images of different people's faces and for each person, there exist ten images with varying facial expression, head rotation, etc. Only the first 10 persons (in the database named "s1" to "s10") were included for the evaluations done here. For each of these people all of the ten corresponding images were used. Hence, the training data comprised 100 instances in total. Each image has a size of 92 by 112 pixels, which corresponds to a 10304-dimensional input vector for M-SOINN. As one person represents one category, the person count refers to the number of categories and person score means category score. The desired counts for clusters and persons were set to 10, due to 10 different people being present in the used dataset. In each scenario 100 runs were performed.

4.3.1 Scenario 1: Random

In this scenario the next input for the algorithm was chosen (uniformly) randomly. As there are 10 different people with ten different images each in the dataset, the probability for picking the same person is 10% and the probability for picking the exact same image is 1%. It is likely for any subsequent inputs to be of a different category, i.e. images from different persons.

By testing different parameter configurations a baseline performance could be established to roughly examine what clustering results are possible without using the modifications. For eight of the ten lowest-scoring results the minimum cluster count was 1, 2 or 3. This means that the corresponding parameter setting allowed M-SOINN in at least one run to generate a topology with 3 or less clusters. Of course, such a small number of clusters is insufficient to represent faces of 10 different people. For all of the ten lowest-scoring results, however, at least one run generated a topology with all 10 people being represented. For the result with the lowest score the generated topologies contained between 10 and 45 clusters. This at least

gave the opportunity to have 10 different people represented in the topology in each run. But the minimum person count was 5 for this parameter setting. Thus, without using modifications none of the ten best parameter settings could guarantee that the topology contained clusters with faces of all 10 different persons. With the parameters $age_{dead} = 100$, $\lambda = 100$, $c_2 = 0.05$, $c_1 = 0.1$ the average number of clusters was 23.5 and the average person count was 8.6, which was the best result (with lowest overall score).

Table 4.4 shows the results obtained with these parameter settings with different modification combinations. The lowest score was achieved when the algorithm used only the modifications EMR, REI and CJ. In this case, the generated topologies always contained at least 9 different persons, mostly even 10 – the average of the number of persons was 9.9. Hence, the algorithm could more reliably store faces of all different people compared to the version without any modifications. But at the same time, the algorithm generated more clusters. In the performed runs the corresponding topologies contained between 20 and 51 clusters, which is at least double as many clusters as minimally required.

NNC	EMR	NMR	REI	CJ	cluster count			person count			cluster score	person score	overall score
					min	max	avg	min	max	avg			
0	**1**	**0**	**1**	**1**	20	51	35.5	9	10	9.9	391036.8	4520.0	1767486336.0
0	0	0	1	1	11	44	27.3	6	10	9.3	43554.0	42000.0	1829268000.0
0	1	0	0	0	15	49	30.8	8	10	9.8	182952.0	10620.0	1942950240.0
0	0	0	0	1	11	39	25.2	5	10	9.0	28101.0	73440.0	2063737440.0
0	1	1	0	1	11	49	31.0	6	10	9.7	68640.0	32000.0	2196480000.0
0	0	0	1	0	16	42	29.7	7	10	9.7	128981.2	20960.0	2703445113.6
0	1	0	0	1	17	49	31.0	8	10	9.7	231897.6	11880.0	2754943488.0
0	1	1	1	1	19	49	32.5	8	10	9.8	291896.0	10530.0	3073664880.0
0	0	1	1	1	14	44	29.4	6	10	9.5	110832.8	36500.0	4045395375.0
0	1	0	1	0	21	50	34.5	8	10	9.8	375937.2	10800.0	4060121760.0
1	0	0	1	1	13	53	38.1	7	10	9.8	209913.4	20000.0	4198268800.0
0	1	1	0	0	17	48	30.2	7	10	9.7	211261.4	20320.0	4292832460.8
0	1	1	1	0	17	50	32.3	7	10	9.9	259618.6	18240.0	4735442534.4
1	1	0	0	1	17	58	40.7	8	10	9.8	522402.7	10710.0	5594933131.2
1	1	0	1	0	27	69	46.3	9	10	10.0	1733140.8	4120.0	7140540096.0
1	0	0	0	0	14	50	31.7	6	10	9.2	172407.1	44250.0	7629011962.5
1	1	0	1	1	31	68	46.5	9	10	10.0	1849650.0	4160.0	7694544000.0
1	1	0	0	0	24	57	40.6	8	10	9.9	773568.0	10170.0	7867186560.0
1	0	0	0	1	13	53	31.3	5	10	9.2	160772.5	65880.0	10591690982.4
1	0	1	0	0	13	58	35.4	6	10	9.2	237842.1	45250.0	10762354120.0
1	0	1	1	1	26	63	44.8	8	10	9.9	1249196.0	9900.0	12367040796.0
1	1	1	0	0	17	66	43.3	7	10	9.8	780900.0	19360.0	15118224000.0
1	0	1	1	0	27	66	45.0	8	10	9.8	1477029.6	10530.0	15553121688.0
1	0	0	1	0	21	53	39.5	6	10	9.8	531257.8	30250.0	16070547240.0
1	1	1	1	1	28	66	50.2	8	10	10.0	1738897.3	9270.0	16119577878.3
1	1	1	0	1	18	67	43.4	7	10	9.9	898884.0	18240.0	16395644160.0
1	0	1	0	1	15	61	36.8	6	10	9.4	407072.6	41000.0	16689978240.0
1	1	1	1	0	26	72	50.7	8	10	9.9	2100059.6	9540.0	20034568965.6
0	0	1	1	0	7	45	29.0	6	10	9.5	1122638.4	38250.0	42940918800.0
0	0	1	0	0	8	47	25.1	5	10	8.7	732336.0	81360.0	59582856960.0
0	0	1	0	1	8	45	25.8	4	10	9.0	688240.8	98490.0	67784836392.0
0	***0***	***0***	***0***	***0***	6	42	24.8	4	10	8.8	967032.0	108780.0	105193740960.0

Table 4.4: Results of the modifications evaluation for the scenario *Random* with the *AT&T faces* dataset. Listed are all results, sorted by overall score in ascending order. The baseline combination (without any modifications) is highlighted in bold italics. The combination chosen for phase 3 is underlined and highlighted in bold.

Examined in phase 3 was an actual topology which has been generated by the

M-SOINN algorithm with the modifications EMR, REI and CJ enabled. Figure 4.4 shows the resulting topology, which contained 37 clusters with all 10 different people being represented. Most clusters consisted of only 2 instances and represented only one variation of a person's face. Also clusters with 3 or 4 instances accounted only for one variation. There existed only four bigger clusters that represented several variations of the corresponding person's face. These clusters stored 22, 10, 9 and 7 instances, respectively. For the latter two clusters it was not possible (due to the number of instances) to include all variations of a particular face. The cluster with 22 instances contained faces of two different persons and should have been split apart into two separate clusters. The algorithm created a topology with more clusters than actually required. Although all of these clusters together contained many face variations of all 10 different persons, only in a few cases different variations ended up in one cluster. In total the topology contained 130 instances, which is more than the number of distinct images.

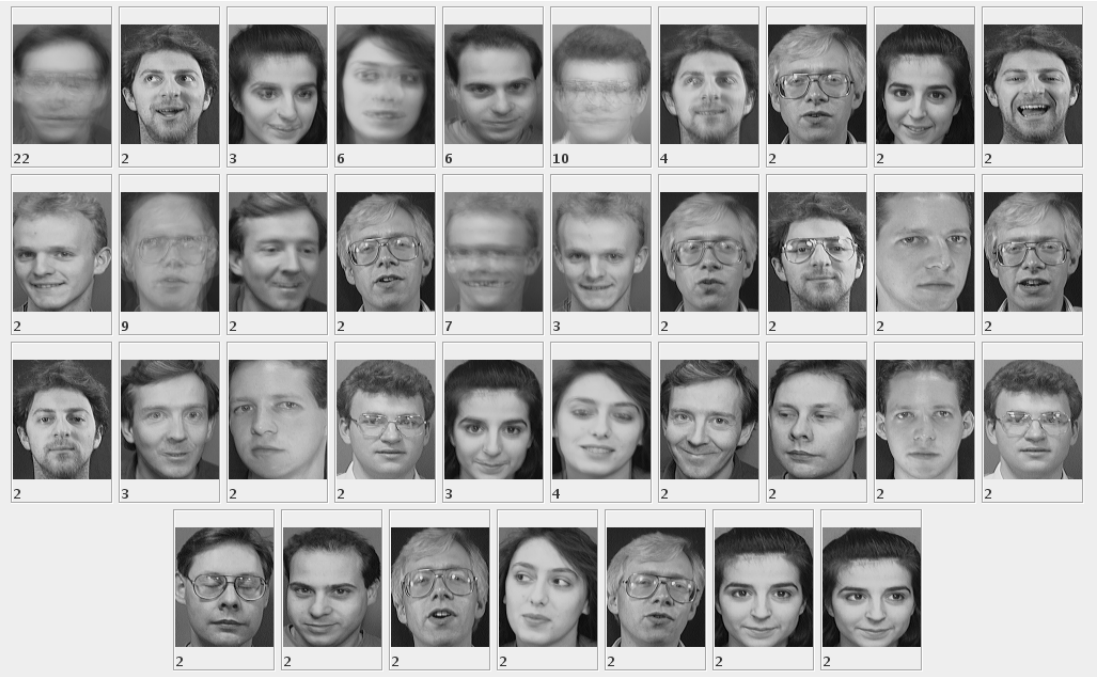


Figure 4.4: Sample topology for the scenario *Random* with the *AT&T faces* dataset. Each rectangle represents a different cluster. The number in each rectangle indicates the number of nodes contained in this cluster and the image visualizes the cluster mean.

4.3.2 Scenario 2: Ordered-Random

In the *Ordered-Random* scenario the inputs were grouped by person which means that the algorithm received various images of a particular person before succeeding with the images of another person. The person order was in accordance with the order of people in the dataset, i.e. at first different faces of person "s1" were shown, then faces of person "s2", etc. During each person's presentation, the next input was

chosen randomly amongst all the variations of this person's face. Hence, in various runs, the overall image order was different. In each run, 1000 inputs were presented in total, which corresponds to 100 inputs per person.

The clustering performance without using any modifications was used as a baseline to compare further results with. The best (lowest) overall score was achieved with the parameters $age_{dead} = 100$, $\lambda = 50$, $c_2 = 0.02$, $c_1 = 0.1$. For this configuration, the generated topologies always represented 10 different people and contained between 35 and 68 clusters. For all other mentioned parameter settings, the cluster counts are in similar ranges between 10 and 25 with averages around 17, and the persons counts range from 8 to 10 with averages around 9.8. Although these topologies could not always represent all 10 people, they consisted of fewer clusters and, thus, obtained a significantly lower cluster score compared to the top result.

NNC	EMR	NMR	REI	CJ	cluster count			person count			cluster score	person score	overall score
					min	max	avg	min	max	avg			
1	**0**	**0**	**1**	**1**	24	64	45.8	10	10	10.0	1243407.0	1.0	1243407.0
0	0	0	1	1	32	65	51.5	10	10	10.0	1860284.2	1.0	1860284.2
0	0	0	1	0	41	69	52.0	10	10	10.0	2393126.4	1.0	2393126.4
1	1	0	0	0	44	70	57.3	10	10	10.0	2783677.1	1.0	2783677.1
1	1	0	0	1	43	72	58.5	10	10	10.0	3179584.8	1.0	3179584.8
1	1	0	1	1	43	74	58.5	10	10	10.0	3502054.4	1.0	3502054.4
0	1	0	1	1	52	75	63.4	10	10	10.0	3705973.9	1.0	3705973.9
0	1	0	0	1	47	76	63.9	10	10	10.0	4193262.0	1.0	4193262.0
1	1	0	1	0	44	78	58.8	10	10	10.0	4211880.8	1.0	4211880.8
0	1	0	1	0	48	79	64.4	10	10	10.0	4837996.8	1.0	4837996.8
0	1	0	0	0	46	83	63.5	10	10	10.0	5669357.6	1.0	5669357.6
1	0	1	0	0	14	44	31.9	8	10	9.6	124178.3	12870.0	1598174077.5
1	0	1	1	1	13	46	32.7	8	10	9.5	119208.1	13500.0	1609309080.0
1	0	1	1	0	18	47	31.7	8	10	9.5	232491.6	13410.0	3117712356.0
1	0	0	0	1	30	61	44.3	9	10	10.0	1231776.0	4200.0	5173459200.0
0	0	1	1	0	23	48	36.5	8	10	9.4	390248.0	14130.0	5514204805.2
0	0	1	0	1	20	52	35.7	8	10	9.4	416292.0	14490.0	6032071514.7
1	0	1	0	1	20	48	31.0	7	10	9.5	273080.0	24640.0	6728689968.0
0	***0***	***0***	***0***	***0***	34	64	50.9	9	10	10.0	1785561.3	4080.0	7285089900.0
0	0	1	0	0	18	49	35.4	7	10	9.3	303667.2	27040.0	8211161088.0
1	0	0	1	0	27	57	45.1	8	10	10.0	966098.9	9180.0	8868787718.4
0	0	1	1	1	25	53	37.7	8	10	9.3	584918.4	15210.0	8896608864.0
0	0	0	0	1	33	71	51.1	9	10	10.0	2442566.9	4040.0	9867970195.2
1	1	1	1	0	27	55	42.0	8	10	9.6	792155.9	12870.0	10195046175.6
1	1	1	0	0	28	56	40.2	8	10	9.6	808504.3	12780.0	10332685465.2
1	1	1	1	1	29	57	42.3	8	10	9.5	925680.0	13410.0	12413368800.0
1	0	0	0	0	30	64	43.5	8	10	10.0	1396279.5	9450.0	13194841275.0
0	1	1	0	0	31	59	44.6	8	10	9.5	1136597.0	13410.0	15241765770.0
0	1	1	1	1	34	57	46.6	8	10	9.4	1082304.0	14310.0	15487770240.0
0	1	1	1	0	36	57	46.1	8	10	9.3	1058935.7	15570.0	16487628537.6
0	1	1	0	1	32	59	45.0	8	10	9.3	1158556.0	15030.0	17413096680.0
1	1	1	0	1	27	59	40.7	7	10	9.6	942678.0	22880.0	21568472640.0

Table 4.5: Results of the modifications evaluation for the scenario *Ordered-Random* with the *AT&T faces* dataset. Listed are all results, sorted by overall score in ascending order. The baseline combination (without any modifications) is highlighted in bold italics. The combination chosen for phase 3 is underlined and highlighted in bold.

Tested in phase 2 was how the algorithm performs when using the introduced modifications with the parameters $age_{dead} = 100$, $\lambda = 50$, $c_2 = 0.02$ and $c_1 = 0.1$. The results for all combinations are shown in Table 4.5. With the ten best modification combinations (according to overall score) M-SOINN achieved in every run to generate

a topology that represents faces of all 10 people. For all these results the minimum and maximum person counts were both 10. Yet, the number of clusters differed depending on the chosen modifications. Amongst these results, the lowest minimum cluster count as well as the lowest maximum cluster count were achieved when using only the modifications NNC, REI and CJ. Corresponding numbers are 24 and 64, respectively. The average of the number of clusters was 45.8. Compared to the baseline performance without any modifications, the algorithm could decrease the number of clusters when using the modification NNC, REI and CJ, while maintaining the same person count. For other modification combinations, either corresponding cluster counts were higher or the minimum person count fell below 10.

For generating a sample topology one run was executed with only the modifications NNC, REI and CJ. The obtained topology contained 38 clusters and faces of all 10 persons were represented (Figure 4.5). Each person's face was represented by at least two and at most seven clusters. For five persons clusters existed with more than 10 instances. For two persons the biggest clusters contained 8 and 9 instances, respectively. Although these bigger clusters included different face variations, in some cases these were only two different similar variations. All other clusters consisted of 6 or less instances with the majority only having 2 instances. The clusters with 2 instances accounted for only one particular variation of the corresponding person's face. In these cases, more clusters were required to account for the various variations. The total number of instances contained in this topology was 155, which is more than the actual number of distinct images.

Figure 4.5: Sample topology for the scenario *Ordered-Random* with the *AT&T faces* dataset. Each rectangle represents a different cluster. The number in each rectangle indicates the number of nodes contained in this cluster and the image visualizes the cluster mean.

4.3.3 Scenario 3: Permuted-Random

Similarly to the scenario *Ordered-Random*, in this scenario the inputs were grouped by person but the person order was randomised. Images of a particular person were presented before succeeding with the images of another person. The images of one person were chosen randomly amongst the variations of this person's face. M-SOINN received 1000 inputs per run which corresponds to 100 inputs per person.

In order to establish a baseline result evaluations were performed for M-SOINN without using any modifications. Considering the 10 best (lowest-scoring) results according to the overall score metric, for almost all these parameter settings both the minimum and maximum person counts were 10. This means that, for these cases, the generated topologies always contained faces of 10 different people. The average of the number of clusters, however, varied significantly for these different parameter settings. The lowest average number of clusters was achieved by the lowest-scoring parameter configuration, which was $age_{dead} = 1000$, $\lambda = 10$, $c_2 = 0.02$, $c_1 = 0.1$. For this configuration the actual number of clusters ranged from 21 to 47 over the various runs.

NNC	EMR	NMR	REI	CJ	cluster count			person count			cluster score	person score	overall score
					min	max	avg	min	max	avg			
0	**0**	**0**	**1**	**0**	23	47	36.0	10	10	10.0	359100.0	1.0	359100.0
0	0	0	1	1	24	50	35.5	10	10	10.0	439866.5	1.0	439866.5
1	0	0	1	0	34	65	47.7	10	10	10.0	1732416.0	1.0	1732416.0
1	1	0	0	0	67	91	79.3	10	10	10.0	8356292.0	1.0	8356292.0
1	1	0	0	1	62	90	78.9	10	10	10.0	8706075.2	1.0	8706075.2
1	1	0	1	0	69	96	83.1	10	10	10.0	10833379.2	1.0	10833379.2
1	1	0	1	1	67	99	83.5	10	10	10.0	12828202.2	1.0	12828202.2
0	0	1	0	1	8	15	11.6	7	10	9.1	3801.6	29920.0	113743872.0
0	0	1	1	0	8	16	11.9	7	10	9.4	5537.7	25280.0	139993056.0
0	0	1	1	1	8	16	11.8	6	10	9.3	5310.9	41500.0	220402350.0
0	0	1	0	0	6	15	11.4	6	10	9.3	7320.0	43750.0	320250000.0
1	0	1	1	1	8	18	13.1	6	10	8.7	12266.1	57250.0	702234225.0
1	0	1	0	0	7	17	12.2	6	10	8.2	11158.4	70500.0	786667200.0
0	***0***	***0***	***0***	***0***	24	47	33.5	9	10	10.0	334612.8	4040.0	1351835712.0
1	0	1	1	0	8	22	13.1	6	10	8.5	23985.0	62500.0	1499062500.0
1	0	1	0	1	6	18	12.2	5	10	8.2	18544.5	99360.0	1842581520.0
0	0	0	0	1	20	45	33.6	8	10	10.0	253075.7	9450.0	2391565176.0
0	1	0	1	1	24	54	38.2	9	10	9.9	611847.0	4560.0	2790022320.0
0	1	0	0	1	24	44	34.2	8	10	9.8	277940.3	10800.0	3001754700.0
0	1	0	0	0	24	47	33.7	8	10	9.7	337759.2	11880.0	4012579296.0
0	1	0	1	0	28	59	39.3	9	10	9.9	922032.0	4360.0	4020059520.0
1	0	0	0	1	27	62	42.2	9	10	10.0	1140907.7	4120.0	4700539641.6
1	0	0	1	1	30	64	46.9	9	10	10.0	1530894.8	4080.0	6246050580.0
1	0	0	0	0	28	55	42.8	8	10	10.0	827398.3	9180.0	7595516577.6
0	1	1	0	0	0	10	1.9	0	6	1.5	1097470.0	3653650000.0	4009771265500000.0
1	1	1	1	0	3	19	9.4	1	6	2.0	2162400.0	2688000000.0	5812531200000000.0
0	1	1	1	1	0	14	6.3	0	7	3.5	3869250.0	2629440000.0	10173960720000000.0
0	1	1	1	0	0	16	6.7	0	8	3.6	5589430.0	2203740000.0	12317650468200000.0
1	1	1	1	1	2	21	8.6	1	6	1.9	5227200.0	2718000000.0	14207529600000000.0
1	1	1	0	0	1	9	5.1	1	2	1.0	1054800000.0	1792800000.0	18910454400000000000.0
1	1	1	0	1	2	8	5.2	1	3	1.1	1096200000.0	2373600000.0	26019403200000000000.0
0	1	1	0	1	0	6	2.0	0	6	1.6	3468850000.0	3634400000.0	126071884400000000000.0

Table 4.6: Results of the modifications evaluation for the scenario *Permuted-Random* with the *AT&T faces* dataset. Listed are all results, sorted by overall score in ascending order. The baseline combination (without any modifications) is highlighted in bold italics. The combination chosen for phase 3 is underlined and highlighted in bold.

In phase 2 all different modification combinations were evaluated with these

parameter settings. Corresponding results are listed in Table 4.6. With the seven lowest-scoring modification combinations, M-SOINN achieved to always represent the faces of 10 different people in the corresponding topologies. Both minimum and maximum for the person count were 10. Considering only these combinations, the averages of the cluster count ranged from 79.3 to 83.5 when the modifications NNC and EMR were used but not the NMR modification. Lower average cluster counts were achieved when not making use of the NNC, EMR and NMR modifications but using the REI modification. In these two cases, the respective average cluster counts were 36.0 and 35.5. The lower average was achieved when using the CJ modification. But minimum and maximum for the number of clusters were lower for the runs without the CJ modification. This must be attributed to differences in the way the topologies were formed for the two configurations. By joining two clusters, the CJ modification can only reduce but not increase the cluster count.

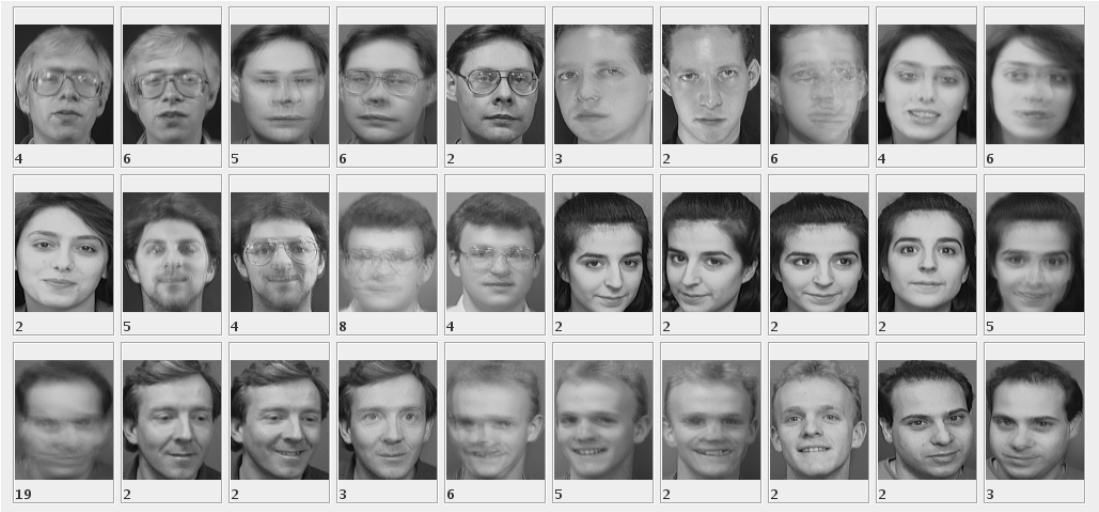


Figure 4.6: Sample topology for the scenario *Permuted-Random* with the *AT&T faces* dataset. Each rectangle represents a different cluster. The number in each rectangle indicates the number of nodes contained in this cluster and the image visualizes the cluster mean.

In one run with the parameters $age_{dead} = 1000$, $\lambda = 10$, $c_2 = 0.02$, $c_1 = 0.1$ and only the REI modification, the M-SOINN algorithm generated a topology with 30 clusters and the faces of all 10 different people (Figure 4.6). In the generated topology many clusters consisted of only 2 instances and represented only one variation of the corresponding person's face. Several clusters with 5 to 8 instances existed which account for two or three variations of a particular face. The topology contained one big cluster with 19 instances. These 19 instances originated from two different persons. The algorithm falsely clustered together inputs from different categories. All other clusters correctly represented only one person and the different face variations were distributed over several clusters. Most people were represented by two or three clusters each, one person's face variations were distributed over five clusters. While storing 126 instances, the topology still could not account for all the presented faces.

4.4 Correlations

The get a better idea of the impact of the different modifications on the resulting cluster count and category count, Pearson's correlation coefficients were computed for respective pairs, using the mean average for each count.

4.4.1 Connection of New Node

For the MNIST digits dataset the NNC modification had a strong influence on the number of clusters. Corresponding Pearson's coefficients in all three scenarios were above 0.98. At the same time, the NNC modification also influenced the digit count, with corresponding correlation coefficients of 0.987, 0.726, 0.631. Hence, on the MNIST dataset the NNC modification led to higher cluster counts and higher category counts. These relationships were not confirmed on the AT&T faces dataset. Pearson's coefficients for the NNC modification and the average cluster count were 0.791, -0.260 and 0.302. The person count and the use of the NNC modification were not strongly correlated; corresponding Pearson's coefficients were between -0.07 and 0.32.

4.4.2 Removal of Longest Edge

Pearson's coefficients for the EMR modification were mostly between -0.4 and 0.4 with only three exceptions for the AT&T faces dataset. For the scenario *Random* the correlation with the average digit count was 0.678, for *Ordered-Random* the correlation with the average cluster count was 0.555 and for the scenario *Permuted-Random* the correlation with the average digit count was -0.516. While none of these coefficients indicate a strong correlation, these values still suggest that the effect of the EMR modification is dependent on the presentation order of the inputs.

4.4.3 Removal of Minimum-Density Node

For the AT&T faces dataset the NMR modification shows mostly a negative correlation with the average cluster count as well as with the average category count. Corresponding Pearson's coefficients were -0.781 and -0.965 for the scenario *Ordered-Random*, and -0.828 and -0.675 for the scenario *Permuted-Random*. Also on the MNIST digits dataset a decorrelation between the average digit count and using the NMR modification was noticeable. Pearson's coefficients were -0.408 and -0.504 for the scenarios *Ordered-Random* and *Permuted-Random*, respectively. When the algorithm made use of the NMR modification, the generated topology was likely to contain less clusters and the total number of represented categories was lower.

4.4.4 Reduction of Local Error

In all but one of the tested cases the REI modification was enabled in the combinations that led to the lowest-scoring results. But corresponding Pearson's coefficients do not suggest that this modification had a strong impact on cluster count or category count. For the MNIST digits dataset the correlation coefficients for the REI modification and the average cluster count were between 0.06 and 0.11. Corresponding coefficients for the REI modification and the average digit count were between 0.074 and 0.295. For the AT&T faces dataset all these coefficients were close to 0 for the scenarios *Ordered-Random* and *Permuted-Random*. Only for the *Random* scenario the coefficient was 0.350 for the REI modification and the average cluster count, and 0.505 for the REI modification and the average person count. Based on these results, it is not possible to deduce a strong relationship between the REI modification and any of these counts.

4.4.5 Joining of Clusters

For all scenarios, Pearson's coefficients for the CJ modification were between -0.3 and 0.3 with most values close to 0. Thus, this modification did not have a strong impact on neither the cluster count nor on the number of represented categories.

4.5 Per-Modification Comparison

This section provides an analysis of the number of clusters and categories, separated by modification. The results obtained without using any modifications are compared with the results when only activating one modification at once. For each modification, both the average cluster counts and the average category counts are examined for all tested scenarios. In particular, it was investigated whether a modification led to an increase or a decrease in the number of clusters or categories. For this purpose, t-tests for unpaired samples assuming unequal variances (Welch, 1947) were performed and corresponding p-values are reported in each case. Observed effects are reported as significant based on a significance level $\alpha = 0.05$.

4.5.1 Connection of New Node

The modification for the connection of new nodes gives M-SOINN the possibility to directly connect a newly created node to the respective nearest node. The modification should lead to a faster formation of clusters and avoid many isolated nodes. As a result, less nodes are removed during the cluster pruning step and more nodes remain in the topology in the form of small clusters. This should result in a higher number of clusters with two or more nodes.

This assumption was confirmed by the results, which show an increased average cluster count when using the modification in all except of one scenario (Figure 4.7). This effect did not occur because of sampling errors (Table 4.7). In five of the six tested scenarios, the number of clusters was significantly higher with the modification enabled than when not using it. A higher number of clusters can represent more variations of the input data, which was reflected in the number of represented categories. The average category count was never lower than without the modification (Figure 4.8). In four cases the number of categories was significantly lower with corresponding p-values below 0.05 (Table 4.8).

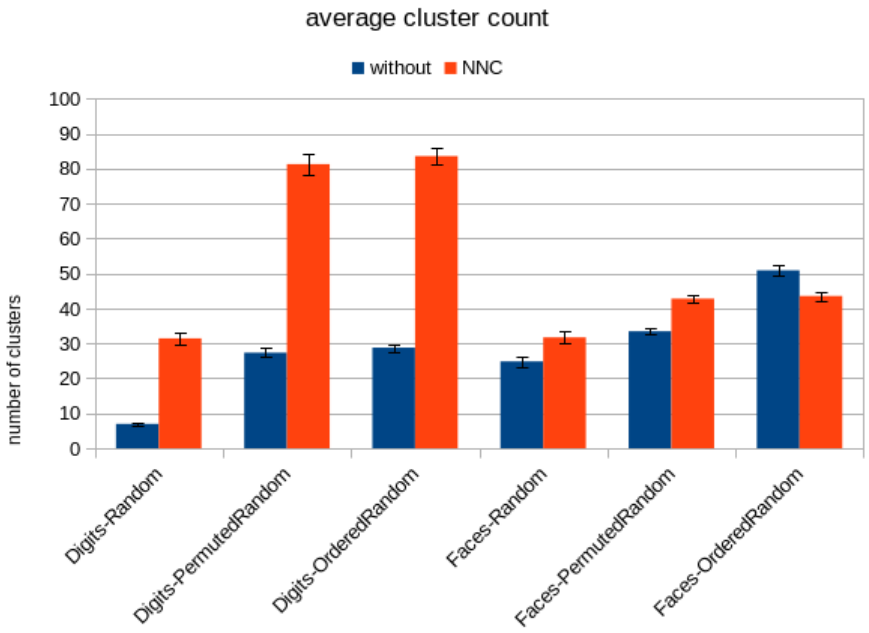


Figure 4.7: Average cluster count for different scenarios without using any modifications and with the NNC modification. Shown are corresponding mean averages (bars) and 95% confidence intervals (1.96 standard error) (lines).

dataset	*Digits*			*Faces*		
scenario	*Random*	*PermutedRandom*	*OrderedRandom*	*Random*	*PermutedRandom*	*OrderedRandom*
p-value	0.000	0.000	0.000	0.000	0.000	0.000

Table 4.7: p-values (rounded) of t-tests on cluster counts obtained without using any modifications and with the NNC modification.

dataset	*Digits*			*Faces*		
scenario	*Random*	*PermutedRandom*	*OrderedRandom*	*Random*	*PermutedRandom*	*OrderedRandom*
p-value	0.000	0.000	0.000	0.009	0.655	0.313

Table 4.8: p-values (rounded) of t-tests on category counts obtained without using any modifications and with the NNC modification.

When having enabled the modification for the connection of new nodes, M-SOINN produced topologies with a higher number of clusters. The cluster counts were higher

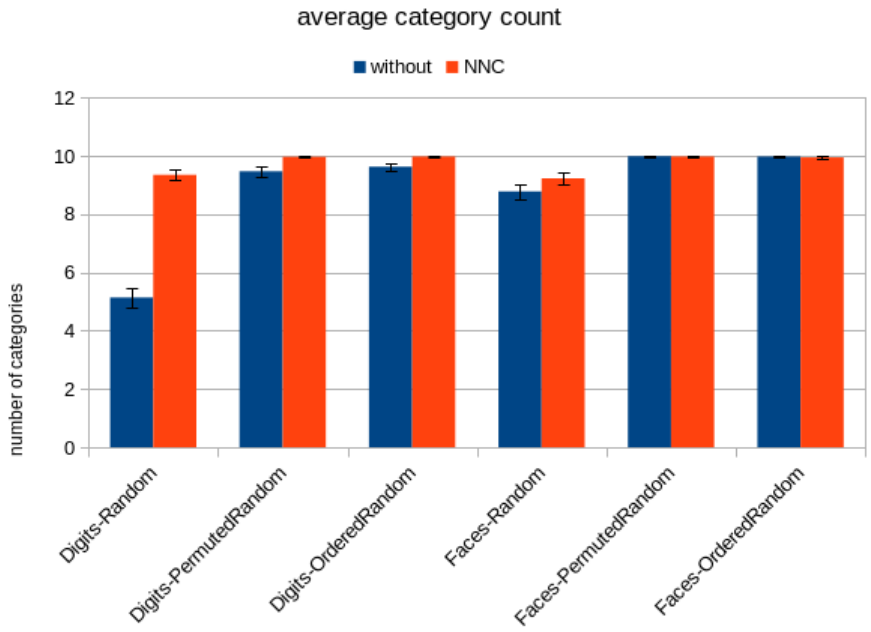


Figure 4.8: Average category count for different scenarios without using any modifications and with the NNC modification. Shown are corresponding mean averages (bars) and 95% confidence intervals (1.96 standard error) (lines).

than the desired number of clusters with average cluster counts above 30. Yet, the generated topologies also represented more input categories, with average category counts above 9. This modification should be enabled when not enough inputs remain in the topology or more input variability needs to be represented.

4.5.2 Removal of Longest Edge

By removing the longest edge in the topology, falsely joined clusters can become separated again. This modification is executed only after a certain number of inputs. As only one edge is removed, the effect on the topological structure should be rather small. The edge removal may increase the number of clusters by one. If the separated nodes are part of a densely connected compound and another route exists between these nodes, no new cluster emerges and the cluster count remains the same.

Indeed, the average cluster counts with and without using the modification are nearly the same in four cases where they differed by at most 1 (Figure 4.9) with corresponding p-values of at least 0.17 (Table 4.9). Thus, in these cases, no statement can be made on how the EMR modification influences the number of clusters. In two scenarios, however, the average number of clusters was significantly higher when using the modification (Figure 4.9), with corresponding p-values below 0.05 (Table 4.9). In four cases the average number of categories increased by at least 0.2 when the modification was used (Figure 4.10). Only in one case, the average category count was lower than the number achieved without the modification. In four cases the

effect on the number of categories is significant with $p < 0.05$ (Table 4.10).

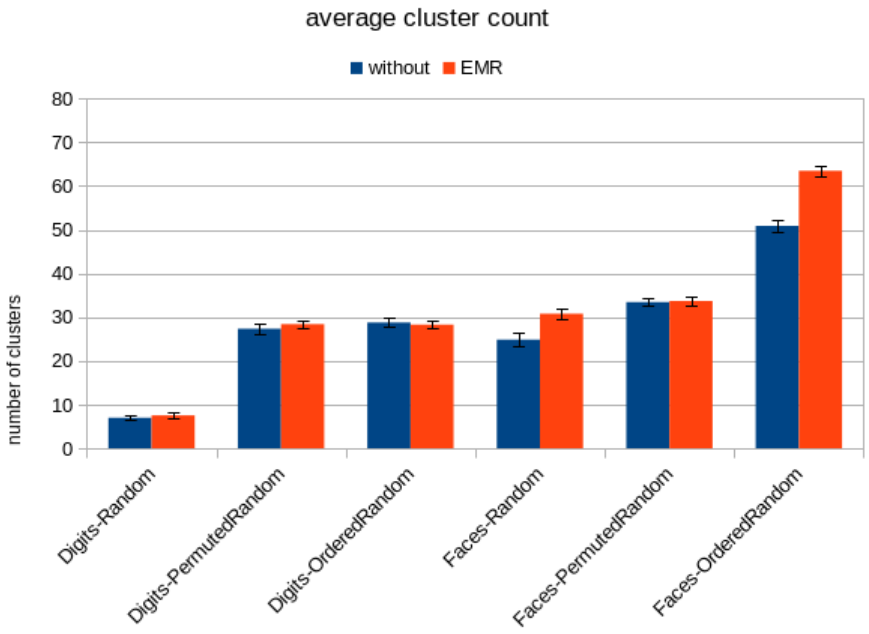


Figure 4.9: Average cluster count for different scenarios without using any modifications and with the EMR modification. Shown are corresponding mean averages (bars) and 95% confidence intervals (1.96 standard error) (lines).

dataset	*Digits*			*Faces*		
scenario	*Random*	*PermutedRandom*	*OrderedRandom*	*Random*	*PermutedRandom*	*OrderedRandom*
p-value	0.170	0.181	0.491	0.000	0.732	0.000

Table 4.9: p-values (rounded) of t-tests on cluster counts obtained without using any modifications and with the EMR modification.

The modification for removing the longest edge did not have a significant impact on the number of clusters in four scenarios but could improve the topology by significantly increasing the number of represented categories in three scenarios. In one case (*Faces-OrderedRandom*) the modification led to a significantly higher cluster count when already M-SOINN without any modifications produced topologies with all categories. This modification is useful if inputs of different categories get merged into single clusters and M-SOINN cannot distinguish between all input categories.

dataset	*Digits*			*Faces*		
scenario	*Random*	*PermutedRandom*	*OrderedRandom*	*Random*	*PermutedRandom*	*OrderedRandom*
p-value	0.065	0.011	0.008	0.000	0.000	0.158

Table 4.10: p-values (rounded) of t-tests on category counts obtained without using any modifications and with the EMR modification.

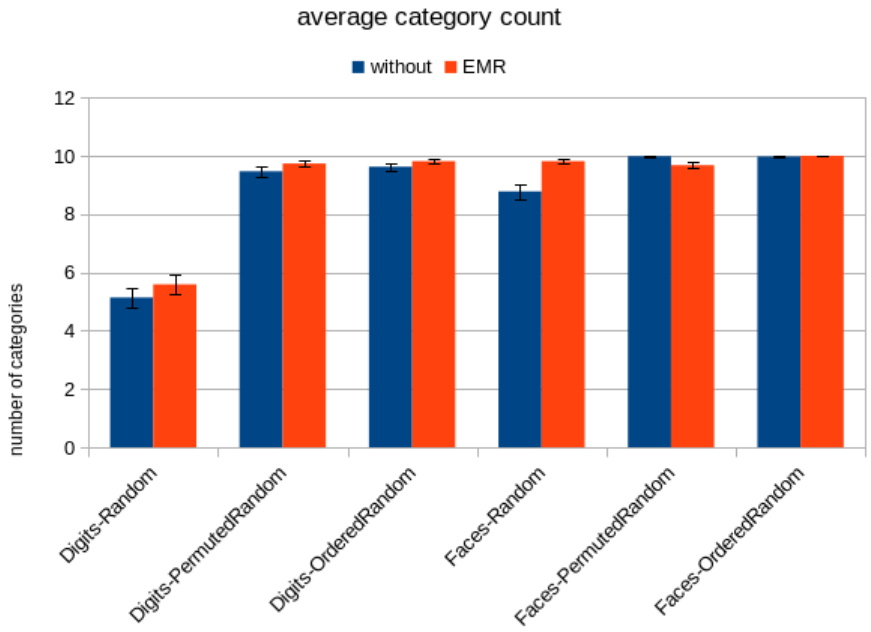


Figure 4.10: Average category count for different scenarios without using any modifications and with the EMR modification. Shown are corresponding mean averages (bars) and 95% confidence intervals (1.96 standard error) (lines).

4.5.3 Removal of Minimum-Density Node

This removal of the node with the lowest number of signals should clean up unrepresentative nodes and lead to a more compact topology. The node count should decrease without reducing the number of represented categories. However, this modification removes only one node in each clean-up step. Thus, the number of clusters should not decrease significantly.

dataset	*Digits*			*Faces*		
scenario	*Random*	*PermutedRandom*	*OrderedRandom*	*Random*	*PermutedRandom*	*OrderedRandom*
p-value	0.396	0.000	0.000	0.843	0.000	0.000

Table 4.11: p-values (rounded) of t-tests on cluster counts obtained without using any modifications and with the NMR modification.

The results show that in two cases the average cluster counts did not differ significantly with $p > 0.05$ (Table 4.11). But in four cases the modification led to a significantly lower number of clusters (Figure 4.11) with p-values below 0.05 (Table 4.11). In these four cases the node removal did significantly reduce the number of represented categories (Table 4.12). Nonetheless, for all scenarios the average category counts were lower with the modification compared to the results without the modification (Figure 4.12). In two scenarios, M-SOINN without modifications achieved average category counts of 10, but lower non-optimal averages when regularly removing the minimum-density node.

With the removal of the minimum-density node, M-SOINN produced topologies

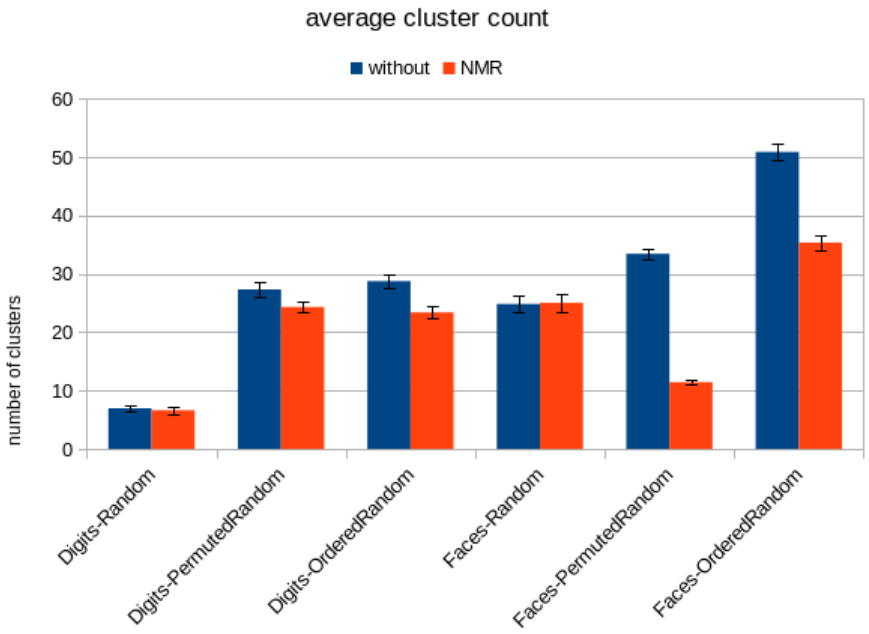


Figure 4.11: Average cluster count for different scenarios without using any modifications and with the NMR modification. Shown are corresponding mean averages (bars) and 95% confidence intervals (1.96 standard error) (lines).

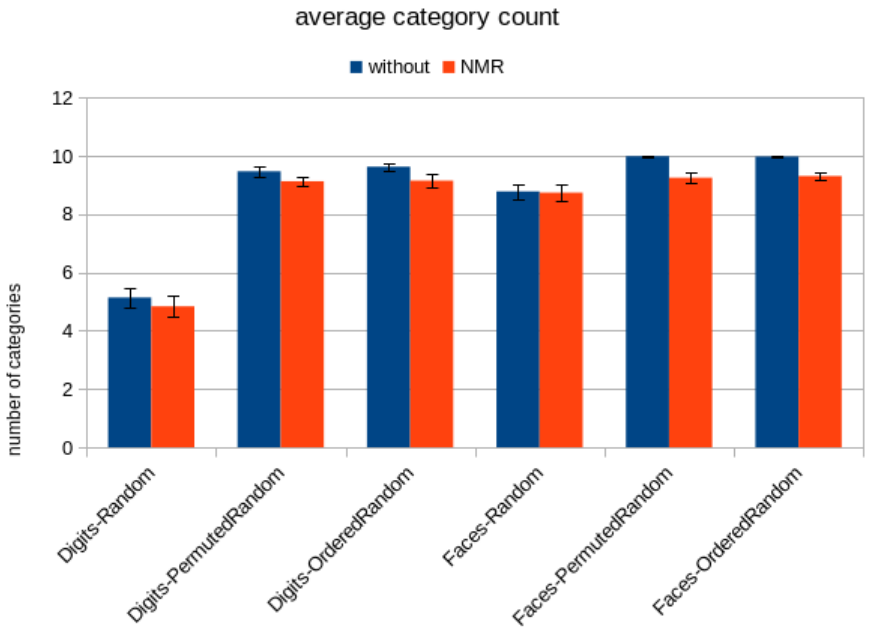


Figure 4.12: Average category count for different scenarios without using any modifications and with the NMR modification. Shown are corresponding mean averages (bars) and 95% confidence intervals (1.96 standard error) (lines).

dataset	*Digits*			*Faces*		
scenario	*Random*	*PermutedRandom*	*OrderedRandom*	*Random*	*PermutedRandom*	*OrderedRandom*
p-value	0.228	0.007	0.000	0.837	0.000	0.000

Table 4.12: p-values (rounded) of t-tests on category counts obtained without using any modifications and with the NMR modification.

with a significantly lower number of clusters in four scenarios. Unfortunately, the removal process also decreased the number of categories in these four scenarios significantly and resulted in certain categories being not represented at all. This modification should be used to reduce the number of nodes and clusters in the topology but should not be activated if M-SOINN fails to represent all input categories.

4.5.4 Reduction of Local Error

The modification for the local error reduction inserts a new node next to the node with the highest accumulated error, in case this node has neighbours. As the new node is directly connected to two already connected nodes, this process does not change the cluster structure of the topology. Instead, the insertion process stabilises the topology in this region. As a consequence, the modification preserves smaller clusters. Yet, the effect on both the cluster count and the number of represented categories should be rather small.

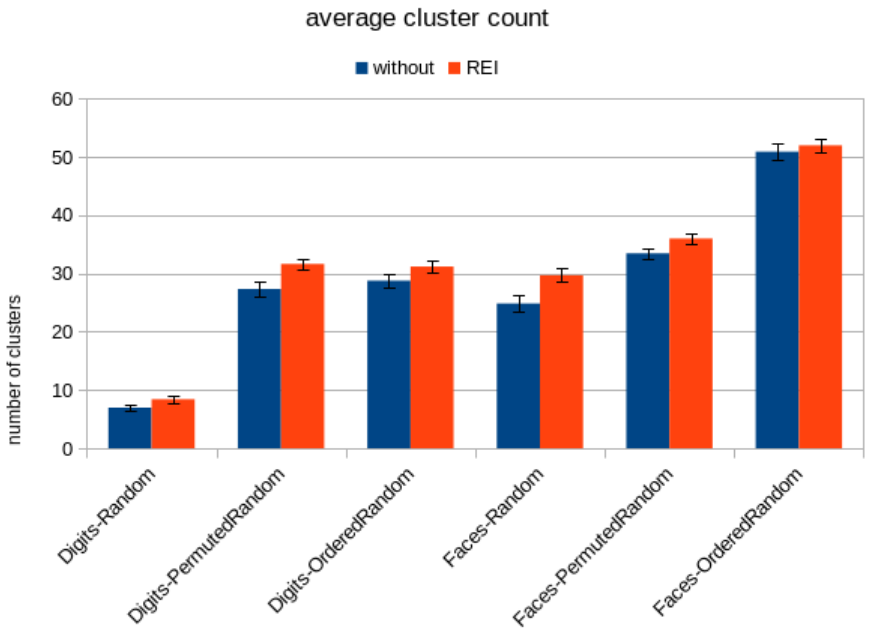


Figure 4.13: Average cluster count for different scenarios without using any modifications and with the REI modification. Shown are corresponding mean averages (bars) and 95% confidence intervals (1.96 standard error) (lines).

dataset	*Digits*			*Faces*		
scenario	*Random*	*PermutedRandom*	*OrderedRandom*	*Random*	*PermutedRandom*	*OrderedRandom*
p-value	0.001	0.000	0.001	0.000	0.000	0.242

Table 4.13: p-values (rounded) of t-tests on cluster counts obtained without using any modifications and with the REI modification.

However, the results showed a significant difference in the number of clusters in five cases with $p < 0.05$ (Table 4.13). When using the modification, the average cluster

counts were significantly higher compared to the results without the modification (Figure 4.13). Only in one case (with $p = 0.242$) this effect is not significant. The increased cluster count allowed M-SOINN to represent more categories in the generated topologies. The average category counts were higher when using the modification (Figure 4.14). This effect was significant in four scenarios with $p < 0.05$ (Table 4.14). In the other two scenarios the maximum average number of categories was already reached without using the modification.

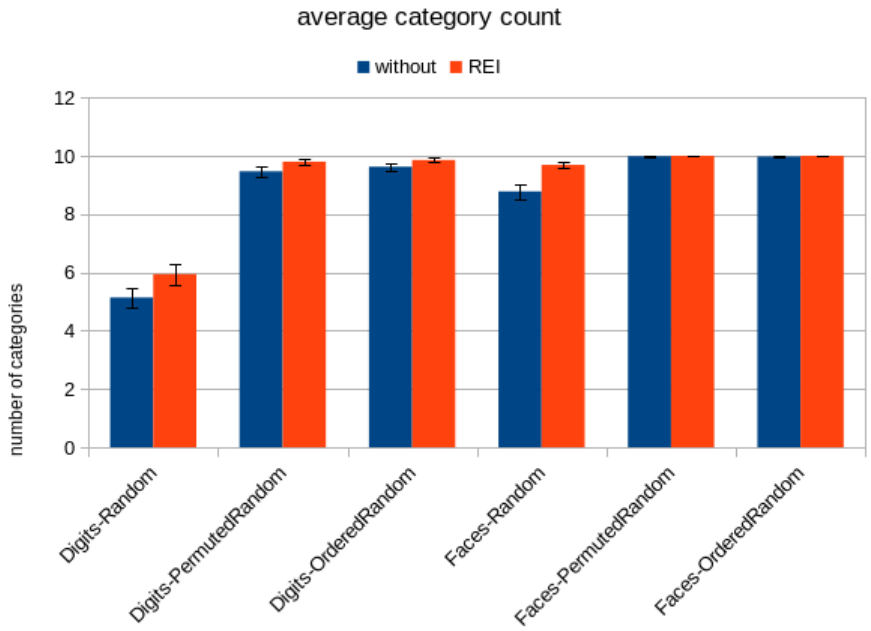


Figure 4.14: Average category count for different scenarios without using any modifications and with the REI modification. Shown are corresponding mean averages (bars) and 95% confidence intervals (1.96 standard error) (lines).

dataset	*Digits*			*Faces*		
scenario	*Random*	*PermutedRandom*	*OrderedRandom*	*Random*	*PermutedRandom*	*OrderedRandom*
p-value	0.001	0.001	0.001	0.000	0.320	0.158

Table 4.14: p-values (rounded) of t-tests on category counts obtained without using any modifications and with the REI modification.

Compared to M-SOINN without any modifications, the regular node insertion process to reduce the local error increased both cluster and category counts. These effects were statistically significant for both counts in four scenarios. However, the respective differences in average cluster counts were at most 5 and the modifications generally improved the cluster structure by avoiding the deletion of weakly represented categories. Hence, this modification is considered useful in many situations and can be activated without any major disadvantages or risks.

4.5.5 Joining of Clusters

The modification for cluster joining can enhance the topology by merging clusters with similar inputs such that a single category is finally represented by a single cluster. In each clean-up step, clusters within a certain distance of each other are joined until no more clusters satisfy the proximity requirement. Obviously, this modification can reduce the number of clusters; at most down to one, such that the topology consists of only one single cluster. However, depending on the topological structure, no clusters or only a few clusters may be joined.

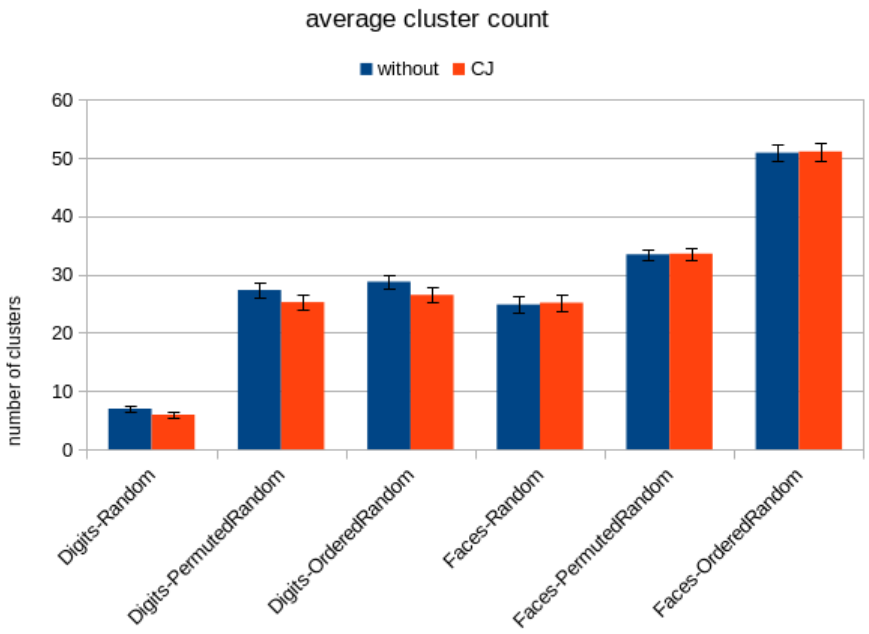


Figure 4.15: Average cluster count for different scenarios without using any modifications and with the CJ modification. Shown are corresponding mean averages (bars) and 95% confidence intervals (1.96 standard error) (lines).

dataset	*Digits*			*Faces*		
scenario	*Random*	*PermutedRandom*	*OrderedRandom*	*Random*	*PermutedRandom*	*OrderedRandom*
p-value	0.011	0.025	0.007	0.767	0.858	0.844

Table 4.15: p-values (rounded) of t-tests on cluster counts obtained without using any modifications and with the CJ modification.

The results show that, for the AT&T faces dataset, the modification did not have a significant effect on the number of clusters (Table 4.15) with average cluster counts differing by at most 0.5 compared to respective counts without using the modification (Figure 4.15). However, for the MNIST digits, the cluster counts were significantly different (Table 4.15) and the modification led to topologies with a lower number of clusters on average (Figure 4.15). The modification could not improve the representation of categories, i.e. lead to more represented categories. Instead, in two scenarios the average category counts were significantly lower (with $p < 0.05$)

with the modification than they were without (Figure 4.16). In one scenario (*Digits-OrderedRandom*) the modification achieved a lower average number of categories and a p-value of 0.075 (Table 4.16) still indicates a very low probably for observing this effect due to a sampling error. For the scenarios with the AT&T faces dataset no significant impact on the number of represented categories can be reported with corresponding p-values above 0.15 (Table 4.16).

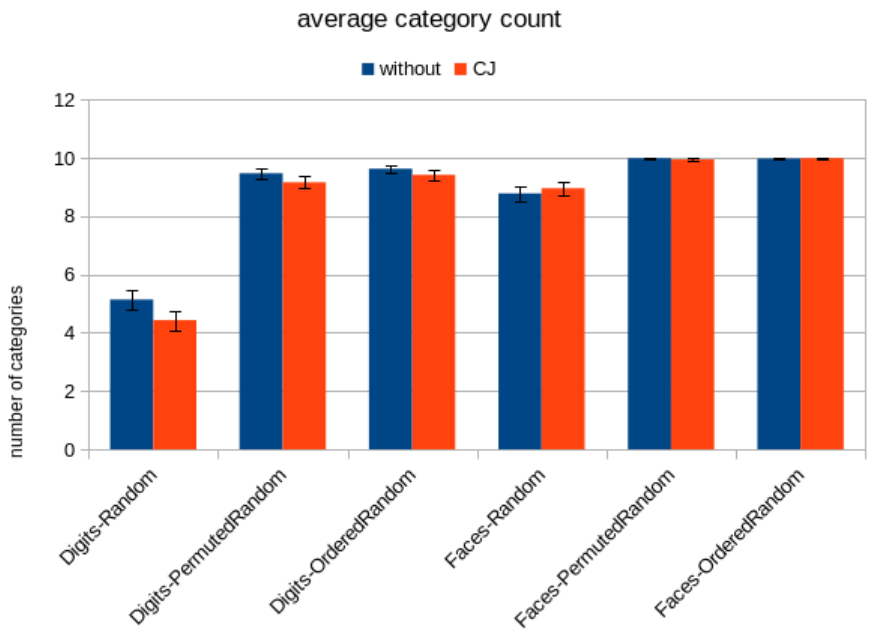


Figure 4.16: Average category count for different scenarios without using any modifications and with the CJ modification. Shown are corresponding mean averages (bars) and 95% confidence intervals (1.96 standard error) (lines).

dataset	*Digits*			*Faces*		
scenario	*Random*	*PermutedRandom*	*OrderedRandom*	*Random*	*PermutedRandom*	*OrderedRandom*
p-value	0.003	0.026	0.075	0.320	0.155	0.563

Table 4.16: p-values (rounded) of t-tests on category counts obtained without using any modifications and with the CJ modification.

While the modification for joining clusters reduced the number of clusters on the MNIST digits dataset, it also reduced the number of represented categories for this dataset in two scenarios. Certain join operations possibly merged two different categories into one cluster, which then prevented the distinction of both categories and resulted in a lower number of represented categories. According to the results, the modification did not prove to be useful. However, the decision whether to join two clusters depends on a join tolerance value. Irrespective of using a relative or absolute join criterion, the corresponding parameter value should be chosen carefully and the optimal value may differ for different datasets.

4.6 Analysis

4.6.1 Clustering Performance

Both with and without using the introduced modifications the M-SOINN algorithm could not achieve the optimal clustering results. The inputs of one category were not all grouped together into one single cluster. Instead, the algorithm created a lot of small clusters that represented only one particular variation of an input category. Bigger clusters could represent several variations but still did not include all the variations of one category. This was obvious in the case of the AT&T faces dataset where the number of distinct inputs per category was exactly ten.

In order to reliably represent inputs of all categories, more clusters than the actual number of categories were needed, which was observed on both datasets. The algorithm never achieved a perfect clustering result where both cluster count and category count exactly correspond to the number of presented categories. Instead, various inputs of one category were distributed over several clusters. In regard to using the generated topologies for further associative learning, having too many clusters is still more desirable than missing to represent certain categories entirely. Instead of forming an association from one cluster, such a relationship can also be expressed by multiple nearly identical associations from different clusters that all represent the same category. On the other hand, if no cluster exists for a certain category, no association can be formed at all with this category.

4.6.2 Modifications

In all tested cases, the algorithm produced better results with lower scores when making use of the modifications compared to the results without any modifications. Above all, the coverage of the different categories was higher when using modifications. Especially important here is that, with the modifications the minimum category count was closer to the desired number of categories, and in all but the *Random* scenarios the desired category count was achieved in all tested runs. Hence, the algorithm with modifications led to topologies that better represented the received inputs. Compared to the algorithm without modifications, the possibility of missing to represent a certain category was a lot lower. However, although each person was represented with at least one face variation, not all of the shown variations were stored in the generated topologies.

Amongst the different modification combinations, good results on both datasets were achieved when the algorithm used the REI modification but not the NMR modification. In the case of the MNIST digits, the lowest-scoring results were generated with additionally the NNC modification enabled. The clustering performance on the AT&T faces dataset was better when M-SOINN made additional use of the CJ

modification. The benefits of different modifications can also depend on the order of presenting the inputs. In five of six cases the best results (i.e. with the lowest score) were obtained with the REI modification enabled and the NMR modification disabled. In four of six cases the algorithm generated the best (lowest-scoring) results without the NMR modification but with the CJ modification. Also in four of six cases, the lowest score was achieved with the NNC modification and without the NMR modification.

Examining the correlation coefficients between the different modifications and cluster/category counts revealed that using the NNC modification led to a higher number of clusters. The NNC modification allows the algorithm to directly connect a new node to an existing one. Compared to isolated nodes, connected nodes not necessarily removed in the next clean-up step. This also helps to retain more different categories in the topology. When using the NMR modification, the algorithm produced topologies with fewer clusters and a lower number of represented categories. The NMR modification removes one node whenever the topology is tidied up. Small clusters are prone to be torn apart by this node removal. The remaining nodes become even more weakly connected (as they lost one neighbour) and, in case they have less than four neighbours, become removal candidates during cluster pruning. Thus, especially in topologies with many small clusters, the NMR modification can reduce the resulting cluster/category counts. The REI modification adds a node next to the node with the highest local error. This process targets already connected nodes and inserts a node between two connected nodes. But the number of neighbours of the affected nodes does not change. Although the overall number of nodes in the topology is increased, the modification does not influence the removal of these nodes during cluster pruning.

The separate evaluation of every modification gave further insights into each modification's effect on the number of clusters and the number of categories. The modification for connecting new nodes led to a significantly higher number of represented categories (observed in four cases) but also increased the cluster count in the topology (observed in all cases). A better coverage of the presented categories could also be achieved with the modification for removing the longest edge (observed in three cases), while this modification did not significantly impact the cluster count (observed in four cases). A significantly lower number of clusters resulted with the modification for removing the minimum-density node (observed in four cases). However, this modification also significantly reduced the category count (observed in four cases). The modification for reducing the local error significantly increased both cluster and category counts (observed in four cases). The cluster joining process significantly decreased the number of nodes and categories (observed in three cases). This modification can be useful when used in conjunction with other modifications. For instance, the removal of the longest edge can counteract the joining of clusters.

Similarly, the reduction of the local error is complemented by the removal of the minimum-density node – one modification adds a node while the other one removes a node. In situations where the formation of bigger clusters is difficult and many small clusters need to be kept, the modification for connecting new nodes can be useful. By allowing a faster growth of clusters, this modification mitigates the effect of the cluster pruning in the clean-up step, which removes many recently created but weakly connected nodes.

4.6.3 Topology Size

In some cases, M-SOINN stored more nodes in its topology than the actual number of distinct inputs. The AT&T faces dataset used for evaluation consisted of 100 different face images (10 persons with ten face variations each). But the examined topologies all contained more than 100 instances. Simply storing every distinct input would suffice to represent the entire dataset and would take up less storage space. But this would not include any clustering of the inputs and prevent M-SOINN to be tolerant to noise in future inputs.

With the goal to group the single inputs into a smaller number of categories, clustering is essential. M-SOINN is very storage intensive and cannot always reduce or compress the amount of information that needs to be stored. But as the algorithm without modifications mostly fails to represent instances of all the presented categories, M-SOINN with the modifications can at least achieve a higher coverage of the data to be learned – despite demanding for high storage requirements.

4.7 Comparison with SOINN and E-SOINN

With SOINN as well as with E-SOINN, Furao and Hasegawa (2006) and Furao et al. (2007) conducted evaluations using images of the AT&T database of faces. In their evaluations, the images have been resized to 23 by 28 pixels and were processed with a Gaussian filter ($width = 4$, $\sigma = 2$) in order to create the input patterns for the algorithms. Both Furao and Hasegawa (2006) and Furao et al. (2007) used images of 10 people from the database. These were of the subjects "s1", "s12", "s3", "s14", "s5", "s6", "s7", "s8", "s9", "s10".

The algorithms were tested in two scenarios. In a stationary scenario the next input was chosen randomly from all available patterns. In a non-stationary the persons were presented in sequential order and for each person the next image was chosen randomly. The algorithms received 1000 inputs per person, which corresponds to 10000 inputs in total. For both SOINN and E-SOINN the parameters were set to $age_{dead} = 25$ and $\lambda = 25$. For E-SOINN additional parameters were set to $c_1 = 0.0$ and $c_2 = 1.0$. After having learned the topology, the recognition rate was established by iterating over all images and using nearest neighbour classification to find the

nearest node in the topology. Then the cluster of this node was determined and if the cluster prototype represented the same person as the current image, this image was considered as being recognised.

The described evaluations were performed with M-SOINN with parameters similarly set to $age_{dead} = 25$ and $\lambda = 25$. It must be noted here that Furao et al. (2007) use c_1 when deleting nodes with two neighbours and c_2 when deleting nodes with one neighbour. In M-SOINN, these two parameters are swapped such that the parameter subscript corresponds to the considered number of neighbours. Thus, for M-SOINN the parameters were set to $c_2 = 0.0$ and $c_1 = 1.0$. Furao and Hasegawa (2006) remain unclear about which node is taken as the cluster prototype. For M-SOINN the node with the maximum number of signals in the corresponding cluster was chosen as the cluster prototype. M-SOINN was evaluated using all combinations of the introduced modifications. The similarity threshold was dynamic and the cluster join tolerance was relative with $\psi = 1.0$. For each combination 100 runs were performed to obtain a more accurate average of the number of clusters.

In the stationary scenario, most modification combinations led to topologies with only a few clusters. The average cluster counts in respective cases were below 3. Closest to 10 was an average cluster count of 11.91 when using the modifications EMR, REI and CJ. For this combination the lowest number of clusters was 1 and the maximum cluster count was 27. For the corresponding topologies only low recognition rates were obtained; during the performed runs the minimum and maximum values were 0.1 and 0.4, respectively. The average recognition rate was 0.1789. This result is significantly worse compared to the results of both SOINN and E-SOINN.

Also in the non-stationary scenario the generated topologies did not contain enough clusters in many runs. The highest average cluster count was 8.75 which was achieved when using the modifications EMR, NMR and REI. In this case the cluster counts ranged from 2 to 21. Although a maximum recognition rate of 0.69 was reached with this combination, the minimum recognition rate was very low with a value of 0.1. The corresponding average recognition rate was 0.29. This result is better compared to the result of the stationary scenario. But still the clustering performance of M-SOINN is worse than the performances of SOINN and E-SOINN.

Further M-SOINN was tested with different parameter settings. In particular, the parameters c_2 and c_1 were varied, testing value combinations of $c_2 \in \{0.01, 0.05, 0.1\}$ and $c_1 \in \{0.1, 0.5, 1.0\}$. The parameters age_{dead} and λ were left unchanged at 25.

For the stationary scenario, the best results were achieved with $c_2 = 0.05$, $c_1 = 0.1$ and using the modifications NNC, NMR and REI. In 1000 runs with this configuration the algorithm produced topologies with cluster counts between 2 and 50. However, only in six runs more than 36 clusters were generated. In the majority of runs the corresponding topology had at most 19 clusters. The average of the number of clusters was close to the desired count with a value of 10.17. Figure 4.17 shows a

histogram of the number of clusters. The average recognition rate was 0.69. Over all 1000 runs the minimum recognition rate was 0.19 but a maximum recognition rate of 0.99 was achieved, which is near perfect.

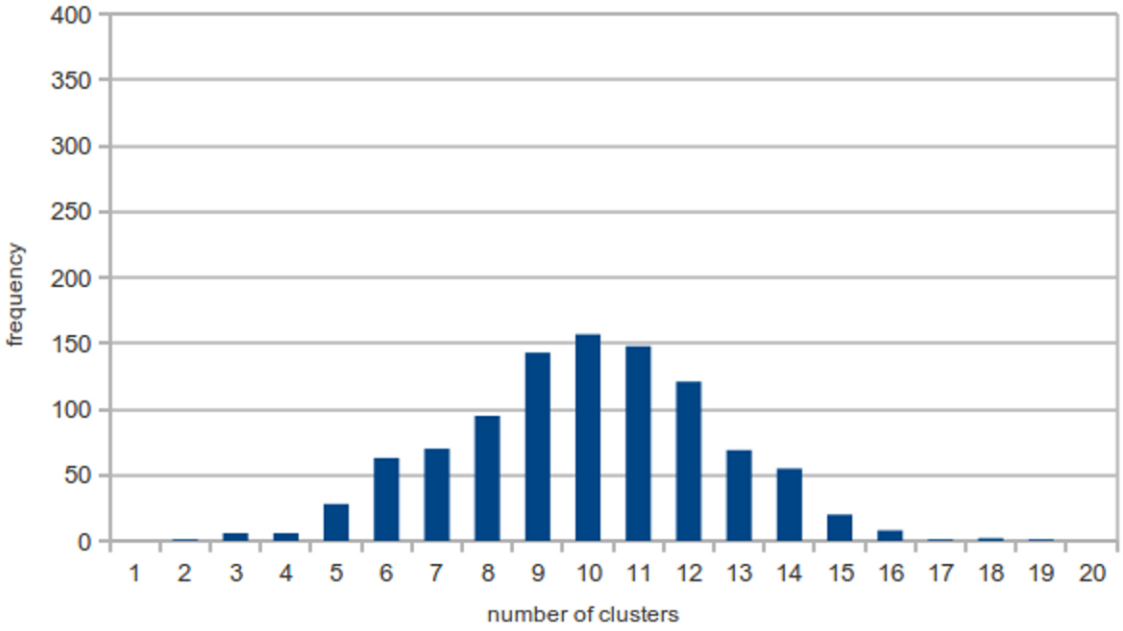


Figure 4.17: Histogram for the number of clusters obtained in the stationary scenario over 1000 runs for the configuration $age_{dead} = 25$, $\lambda = 25$, $c_2 = 0.05$ and $c_1 = 0.1$ when using the modifications NNC, NMR and REI.

In the non-stationary scenario, using the parameter setting with $c_2 = 0.1$, $c_1 = 0.1$ and with the modifications NNC, NMR and CJ led to the best results. Over 1000 runs the number of clusters ranged from 5 to 18 with an average of 9.85. Corresponding frequencies of different cluster counts are shown in Figure 4.18. As in the stationary scenario, again a maximum recognition rate of 0.99 was achieved. But in the non-stationary scenario the minimum recognition rate of 0.37 was higher, as was the corresponding average with a value of 0.82.

In both scenarios M-SOINN can resemble the performance of SOINN but cannot surpass E-SOINN. The cluster count distributions achieved with M-SOINN are similar to the corresponding distributions for SOINN. However, the distributions of E-SOINN are thinner and have higher peak frequencies (Furao et al., 2007). The recognition rates reported for SOINN are 0.90 in the stationary scenario and 0.86 in the non-stationary scenario (Furao and Hasegawa, 2006). These recognition rates are also reported for E-SOINN (Furao et al., 2007).

4.8 Summary

A perfect clustering result was not achieved for any of the tested cases. But the evaluations revealed that, with no modifications, the algorithm tended to underes-

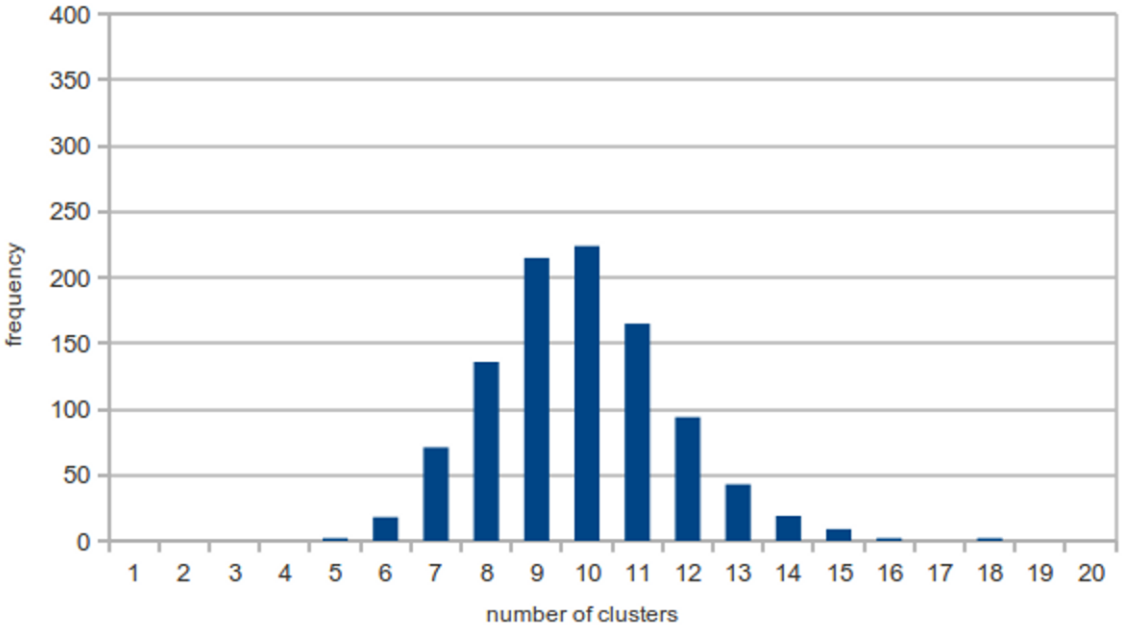


Figure 4.18: Histogram for the number of clusters obtained in the non-stationary scenario over 1000 runs for the configuration $age_{dead} = 25$, $\lambda = 25$, $c_2 = 0.1$ and $c_1 = 0.1$ when using the modifications NNC, NMR and CJ.

timate the actual number of categories that are present in the data and produced topologies with fewer clusters than minimally required. The introduced modifications could always improve the clustering performance and, when using modifications, the algorithm generated better topologies where variations of all categories were represented. The different variations of each category were not always clustered into one single cluster but were distributed over several clusters.

For most modification combinations, the number of clusters was always at least as high as the number of presented categories. Some modification combinations led to extremely high cluster counts. In general, M-SOINN must be considered as a storage-demanding algorithm. But this demand can be alleviated by choosing appropriate modification combinations.

It was not possible to determine a single combination of modifications that always led to best (lowest-scoring) results in all tested scenarios. Nevertheless, the evaluations revealed that the modifications REI and NNC produced good results in the majority of cases whereas the modification NMR seemed to have a negative impact on the clustering performance. The CJ modification proved to be useful only in certain scenarios. The EMR modification was mostly not required to achieve good results although it allows the algorithm to recover from falsely connected clusters.

An analysis of the correlations between the various modifications and cluster counts showed that the NNC modification led to higher cluster counts and, in turn, to a higher number of represented categories. Using the NMR modification results in topologies with a lower number of clusters which also limits the number of categories.

The other modifications did not seem to have a strong influence on the resulting cluster or category counts. The per-modification comparisons gave further insights into how each modification alone effects the number of clusters and categories. While the adequacy of most modifications depends on the particular dataset, the REI modification seems to be generally helpful and should be used whenever possible.

Another series of runs was conducted to evaluate the performance of M-SOINN directly against the performances of SOINN and E-SOINN. In comparison to the latter two the clustering performance of M-SOINN is inferior when using exactly the same values for shared parameters. When choosing other parameter settings for M-SOINN, the clustering performance can be up to par to the standards set by SOINN but cannot outperform E-SOINN.

Chapter 5

Evaluations on Associative Learning

In order to explore the basic functionality of TOSAM, tasks involving associative learning and recall have been tested with the model. First, the model learned certain stimuli and their correlation by exposing it to the corresponding material. Afterwards, the model was given a cue and needed to recall the associated information. The material consisted of different distinct stimuli without any shared or overlapping features amongst them, i.e. the stimuli were meant to be independent of each other. Hence, a symbolic representation was chosen and the inputs were represented by the characters A to J, which also served the purpose of being able to easily distinguish them.

5.1 Symmetric Associations

TOSAM is capable of storing and correlating perceived stimuli using associative learning. Once stored, the model should be able to recall a full concept when only part of its features are available (pattern completion) as well as to retrieve stored information that is related to the perceived input (inference). These tasks address the ability of the model to recall related information, which is achieved by spreading activation over the network's connections.

In this section the effect on recalling groups of associated stimuli is investigated. Associations with different numbers of stimuli involved are trained and then, by giving one of the stimuli as a cue, the activation levels in the other units are measured. The number of associated stimuli ranged from 3 to 10 and, for learning, the stimuli were shown together to the network for a certain time. In each case the presentation time spanned 100 cycles, which resulted in all corresponding connections being almost fully trained with a weight value of 0.982. Thereafter no inputs were shown for 1000 cycles in order to allow the units' activations to decay to a small value near 0.

Finally one of the previously shown stimuli was presented as a cue for 10 cycles and the maximum activation simultaneously achieved by the other units was measured. The reported level was the minimum amongst all non-cue units in a single cycle. However, this value could have been highest after several cycles (and not necessarily directly after cue removal). Considered were 100 cycles after cue removal, and the respective maximum value is reported as the maximum activation, which can be seen as the strength of recall.

Figure 5.1 shows the maximum activation levels of non-cue units measured after cue removal for the tested groups of associated stimuli. A strong recall was achieved for smaller groups but then the recall gradually becomes weaker as the group size becomes larger. For groups of 3 and 4 stimuli the measured activation level in non-cue units was very high with a value of 0.997. However, not all units achieved this activation level at the exact same time. The activation swapped from one unit to another and back. This effect lead to an oscillatory pattern over all units in the network. Such an oscillation was not observed for larger groups. Apart from not being recalled synchronously, also the maximum activation values of the associated units differed. In a group of 5 stimuli all units achieved an activation level of at least 0.762. The recall strength decreased for bigger group sizes. The more stimuli that were associated with each other, the weaker the recall became of non-cue units. In a group of 10 stimuli the maximum activation level of each non-cue unit was only 0.291.

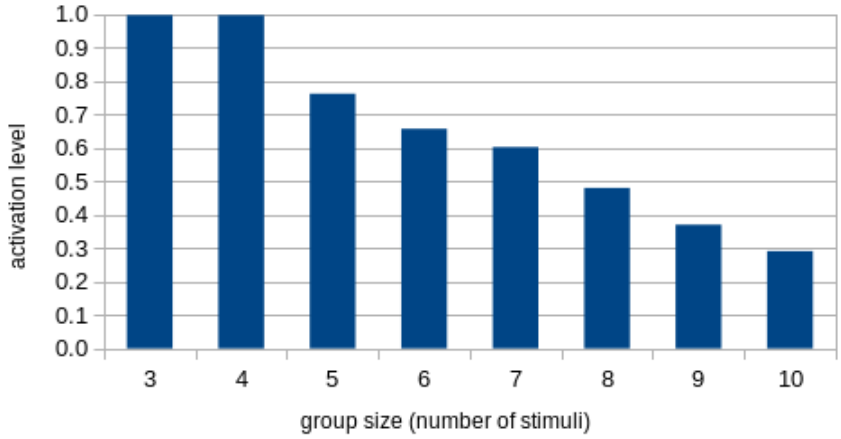


Figure 5.1: The maximum activation level of non-cue units achieved for fully associated groups from 3 to 10 stimuli with 1 stimulus presented as a cue for 10 cycles.

TOSAM allows a unit's activation to accumulate over several cycles. This suggests that a longer cue presentation time could lead to a stronger recall in non-cue units. To evaluate this assumption, another series of runs was performed. Here a group of 10 stimuli was trained exactly as before, and after a gap of 1000 cycles a single stimulus was presented as a cue. The duration of the cue presentation was varied

between 10, 20, 30, ..., 100 cycles. The measured activation levels in non-cue units are shown in Figure 5.2.

Indeed, longer cue presentation times resulted in a stronger recall of non-cue stimuli. Already when presenting the cue for 50 cycles, the activation level in non-cue units was higher than 0.7. But the effect is limited to a certain maximum activation. With an increasing cue presentation time the activation level of non-cue units converged to a value of 0.71. This value was measured for 70 cycles or more of cue presentation. Nonetheless, in these cases the non-cue stimuli can be considered as recalled.

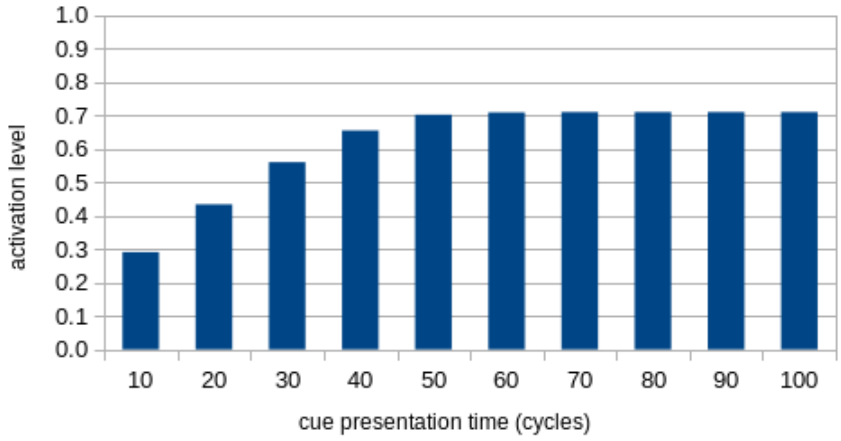


Figure 5.2: The maximum activation level in non-cue units achieved for a fully associated group of 10 stimuli with 1 stimulus presented as a cue for different numbers of cycles.

Another option to increase the strength of recall is to increase the number of stimuli that are presented as a cue. This was tested in another series of runs with cue sizes ranging from 1 to 10 stimuli. The total group size was always 10 and the cue presentation time remained at 10 cycles. The learning procedure was carried out as in the previously described runs.

As can be seen in Figure 5.3, the achieved activation levels in non-cue units increased for cue sizes from 1 to 3. With 3 cue stimuli activation levels of 0.76 were measured in non-cue units. The activation levels for cue sizes of 4, 5 and 6 stimuli were all lower than this value. Presenting 6 cue stimuli resulted in a maximum activation level of 0.64, which was lower than with a smaller cue size of only 3 units. This was a result of activation being spread back to cue units. Once these units had lost activation, they attracted activation by themselves. Any amount of activation that was spread to cue units was not available in non-cue units. For a cue size of 3 this amount was smaller than for a cue size of 6. When giving 70% or more of the learned stimuli as recall cues, a strong recall with activation levels above 0.98 was achieved. In this case, the total amount of activation created in the cue units was enough to highly activate all non-cue units. Combined with the sigmoidal signal

processing, this amount was further amplified and activation levels above 0.98 were achieved.

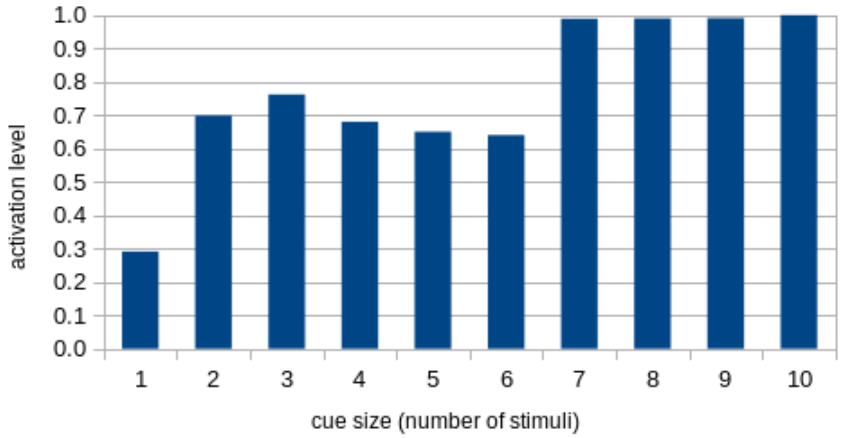


Figure 5.3: The maximum activation level in non-cue units achieved for a fully associated group of 10 stimuli with different numbers of stimuli presented as a cue for 10 cycles.

5.2 Asymmetric Associations

The concept of association can be extended to sequentially perceived inputs. When experiencing first one piece of information and then another, by repeated exposure to this sequence one can assume that the second bit usually follows the first; one can expect the latter upon receiving the first as input, similarly as a conditioned stimulus is followed by a conditioned response in terms of classical conditioning (Anderson, 1999). Both stimulus and response are referred to as conditioned here because they need to be trained; there are no previously built-in unconditioned relationships stored in the network. However, once a correlation between a stimulus and a response has been created, from then on these can be considered as unconditioned. A new input can then be added to create another pairing with either the stimulus or directly with the response. In situations when one stimulus is followed by another (the response), the network should strengthen associations from the stimulus to the response but not vice versa. This is what happens in TOSAM during learning and in accordance to this principle the weights are updated.

Extending the presentation of inputs to more than two stimuli, a chain of associations is formed corresponding to the temporal order in which the inputs were perceived. By presenting different stimuli in a sequential order, TOSAM is able to create associations in a way that allows to recall these sequences in their correct order. This means, once given a stimulus of a sequence as a cue, the unit corresponding to that cue stimulus will be activated and activation will spread from one unit to another based on the unidirectional associations formed during learning. Looking at

only the respective maximally activated unit, the sequence is recalled stimulus by stimulus. To test this functionality different scenarios as well as related parameters have been investigated.

In a first series of runs sequences of different lengths have been presented to the network to form associations between corresponding units. The characters "A" to "J" were used as stimuli and were always presented in their correct alphabetical order. Each stimulus was given as an input for 10 cycles, followed by a gap of 2 cycles. In every sequence each stimulus was contained exactly once, i.e. no stimuli were repeated. After one sequence presentation, a gap of 100 cycles allowed the input loads to decrease, such that no association was created from "J" to "A". This process was repeated 25 times, meaning that each sequence was shown 25 times. After a gap of 1000 cycles the activation levels in all units had decayed to near 0. Then the first stimulus of the sequence was presented as a cue for 1 cycle and the activation in the last unit was measured. Using this setup, associations to subsequent stimuli, e.g. from "A" to "B", were nearly fully trained with weights of ca. 0.986. Connections in the opposite direction developed negative weights, e.g. the connection from "B" to "A" had a weight of -0.098 and the connection weight from "F" to "A" was -0.454. Also backward-facing associations between stimuli that are further spaced apart in the sequence had this weight value. No backward-facing connections had stronger negative weight values. Also sequentially forward-facing connections to indirectly following stimuli developed negative weights, e.g. the weight from "A" to "C" was -0.157 and the connection from "A" to "F" had a weight of -0.444.

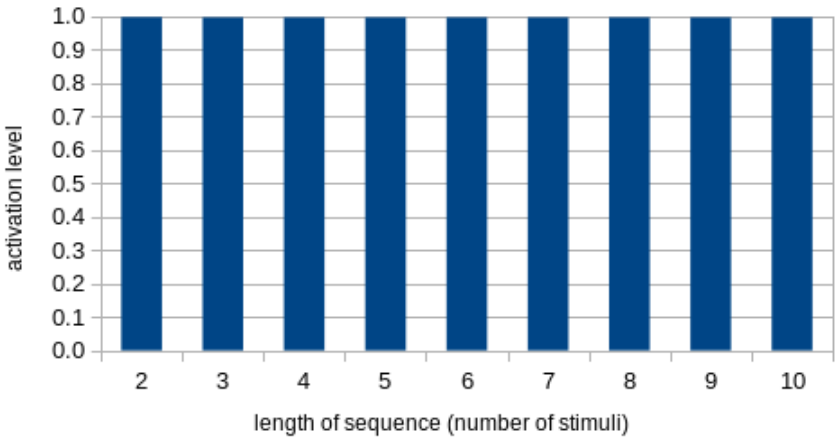


Figure 5.4: The maximum activation level measured in the last unit for sequences of different length when learning each sequence 25 times with a gap of 2 cycles after each stimulus presentation and presenting the first stimulus of the corresponding sequence as a cue for 1 cycle.

This scenario allowed a strong recall of the last stimulus for all tested sequences (Figure 5.4). For each sequence length, the maximum activation level in the respective last unit was 0.996. This value was measured after a specific number of cycles after

cue presentation. This delay depends on the sequence length as the activation needs to be spread from unit to unit until finally reaching the last unit. For a sequence of 10 stimuli the spreading process is visualized in Figure 5.5. It can be clearly seen how the activation was propagated from one unit to another along the learned sequential order. When presenting the recall cue "A", an activation of 1.0 was generated in the corresponding unit. This was the total amount of activation available in the network as the cue was directly removed again after 1 cycle. Nonetheless, this amount was enough to highly activate every unit of the learned sequence. The activation decay in each unit was counteracted by the signal strengthening during the spreading process. Also a sequence of 20 stimuli could be strongly and clearly recalled.

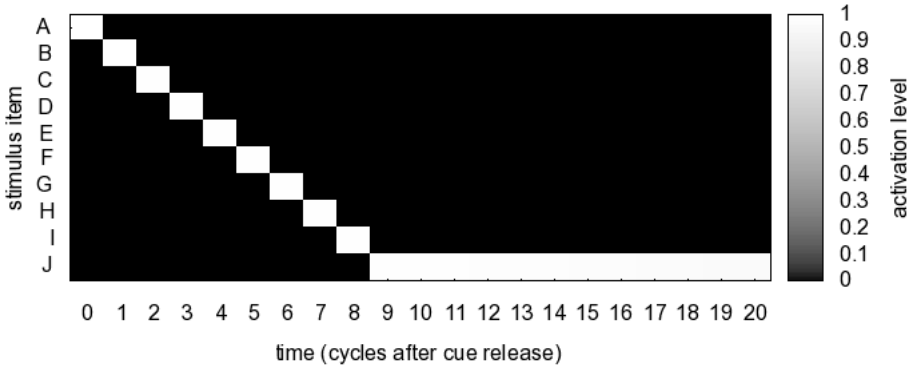


Figure 5.5: The activation levels in all units over several cycles after cue removal when the network learned the 10-stimulus sequence "A" $\rightarrow$ "B" $\rightarrow$ "C", ..., "J" 25 times with a gap of 2 cycles after each stimulus presentation and the cue "A" was shown for 1 cycle.

In another series of runs the same sequences as tested above were presented only 10 times during training. Furthermore, there was no gap between the stimuli, i.e. the release of one stimulus was directly followed by the presentation of the next stimulus. Other parameters were left unchanged. Under these circumstances the connection weights to directly following stimuli were still trained with a weight value of ca. 0.914. However, the effect of unlearning was not as strong as in the previous scenario, e.g. the connection from "F" to "A" had a weight of -0.214 and the association strength from "C" to "A" was -0.116. The connection from "B" to "A" had a positive weight value of 0.316. Also the association strength from "A" to "C" was positive with a value of 0.161.

Figure 5.6 shows the maximally achieved activation levels in the respective last units for different sequence lengths. Only very short sequences could be recalled strongly until the last unit. For a sequence of 4 stimuli, the maximum activation level in the last unit was 0.711. For all longer sequences, the respective activation values were below 0.5. There was almost no recall of the last stimulus for sequences of 8 or more stimuli. For a sequence of 10 stimuli, the spreading of activation is

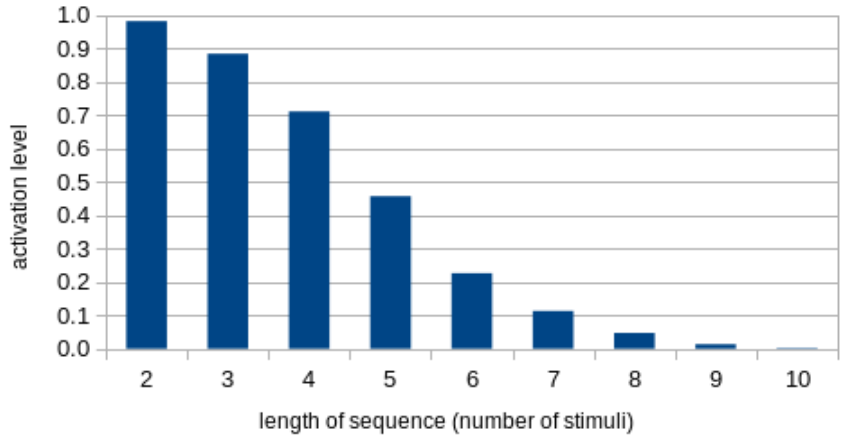


Figure 5.6: The maximum activation measured in the last unit for sequences of different length when learning each sequence 10 times without any gap after each stimulus presentation and presenting the first stimulus of the corresponding sequence as a cue for 1 cycle.

shown in Figure 5.7. The sequential pattern is visible for the first 5 stimuli but then becomes hardly recognisable as the activation levels of all units become similarly low. The total amount of activation was distributed over the first few units and only a small amount reached the remaining units. Such a spreading behaviour is insufficient for a clear recall of the entire sequence. If the trained weight configuration does not represent the sequential order clearly enough, even short sequences cannot be recalled strongly and longer sequences neither entirely.

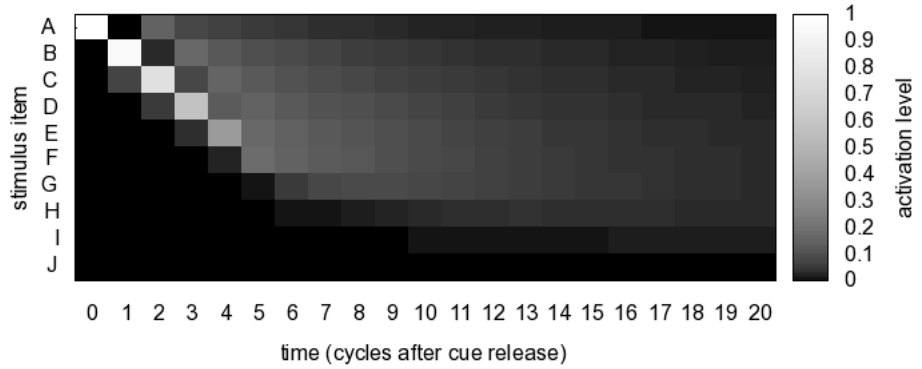


Figure 5.7: The activation levels in all units over several cycles after cue removal when the network learned the 10-stimulus sequence "A" → "B" → "C", ..., "J" 10 times without any gap after each stimulus presentation and the cue "A" was shown for 1 cycle.

A possibility for allowing a stronger recall of all stimuli is to provide more activation to the network. This can be achieved by presenting the recall cue for more cycles. Giving cue input for more cycles allows the activation levels of further units to fill up over time and extend the range of units to which the spreading process

reaches. The previously described run for a sequence of 10 stimuli was repeated but with different cue presentation times. Each sequence was trained 10 times and each stimulus was shown for 10 cycles without any gap between subsequent stimuli. After each entire sequence, 100 cycles without any input allowed the input loads to decrease. Once learning was completed, a gap of 1000 cycles followed and the first stimulus was given as a recall cue. In 10 different runs the cue was shown for 1, 2, 3, ..., 10 cycles, respectively.

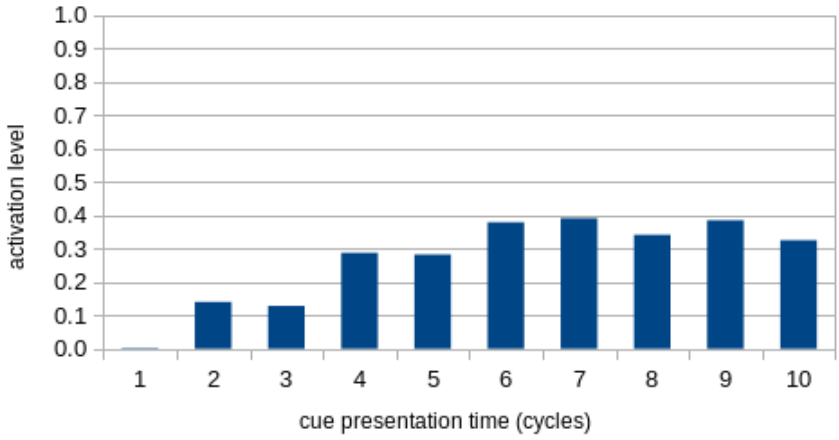


Figure 5.8: The maximum activation measured in the last unit of a sequence of 10 stimuli when learning the sequence 10 times without any gap after each stimulus presentation and presenting the first stimulus as a cue for different numbers of cycles.

Although the maximally achieved activation level in the last unit was generally higher for longer cue presentation times (e.g. each activation level for 6 to 10 cycles was higher than any level for 1 to 5 cycles), the achieved activation levels were all below 0.5 (Figure 5.8). The longest presentation time did not lead to the highest measured activation. Presenting the recall cue for 10 cycles activated the last unit up to a level of 0.325 whereas an activation level of 0.391 was achieved with only 7 cycles. In no case the maximum was reached directly after the cue had been removed. Instead, it took several cycles for the activation to build up (from 9 to 14 cycles for the tested cue presentation times). For longer cue presentation times, activation could already propagate through the network while the cue was still shown. Additionally, activation could spread back to previous units (i.e. against the sequential order). The amount of activation spread by each unit was further influenced by the activation level of the respective destination unit (signal attraction) and by the sigmoidal signal processing. As a result of these spreading dynamics, the maximum activation level in the last sequence unit did not always increase with the number of cue presentation cycles. Figure 5.9 shows the development of the activation levels in all 10 units for a cue presentation time of 10 cycles. The spreading of activation followed the correct sequential order but the sequence was initiated at different times. At various specific moments, several units were highly activated at once and the recall did no longer

happen strictly unit by unit. The resulting activation pattern was an overlap of different stimuli. Over time, the activation levels in all units decayed. Only the unit corresponding to the last stimulus "J" remained at a reasonably high activation level that also decayed slowly.

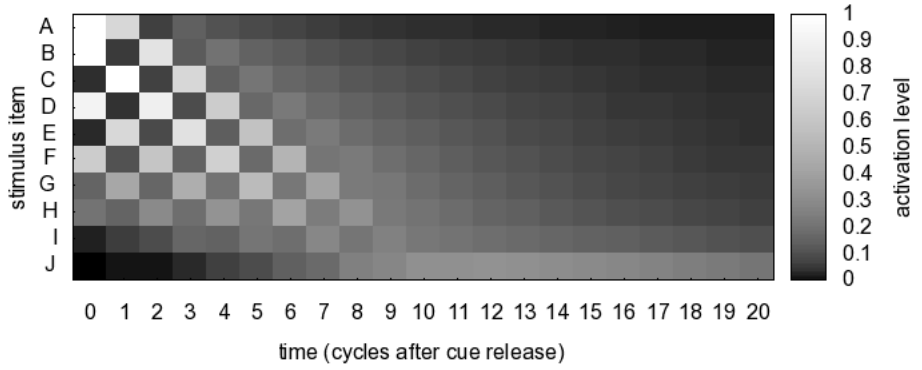


Figure 5.9: The activation levels in all units over several cycles after cue removal when the network learned the 10-stimulus sequence "A" → "B" → "C", ..., "J" 10 times without any gap after each stimulus presentation and the cue "A" was shown for 10 cycles.

5.3 Analysis

5.3.1 Recall Performance for Symmetric Associations

TOSAM could recall strongly associated stimuli in different quantities. But the recall became weaker as the group size became larger. During the recall process only a certain amount of activation became available in the cue unit and could be spread to non-cue units. The amount of activation a unit can send is divided into smaller amounts. These amounts become smaller the more units there are to be reached.

To strengthen the recall for larger groups of stimuli, a successful option was to present the recall cue for a longer time. The units could reach higher activation levels by accumulating small amounts over a longer period. In this way, the total amount of activation in the network could increase over time. However, the recall strength could only be increased to a certain limit. After having reached this limit, the activation decay in all non-cue units could no longer be compensated.

Another possibility was to increase the number of stimuli used as recall cues. Already a cue size of 20% of the stimuli resulted in an improved recall performance but for a strong recall at least 70% of the stimuli needed to be presented. This is already a major amount of the associated units and only a few stimuli remain to be recalled. More desirable is a strong recall with only a small fraction of the stimuli

provided as a recall cue. A combination of an extended cue presentation and the use of multiple recall cues should lead to a strong recall of the entire group.

5.3.2 Recall Performance for Asymmetric Associations

Sequentially trained stimuli could be fully recalled when the corresponding associations had been trained clearly and strongly. This was achieved by a small gap between the presentations of all stimuli, and by the frequent repetition of the sequences. The temporal order in which the stimuli had been learned was clearly reflected in the order of retrieval. Any negatively trained weights helped by inhibiting the recall of out-of-sequence stimuli.

The formation of negative weights was caused by the extinction part of the learning rule (Equation 3.6). If, for a given connection, the destination unit had an input load below $0.8/1.8 \approx 0.44$, its contribution towards the weight update was negative. In particular, this means that whenever a single stimulus was perceived alone, the weight values of connections from all other units with input load decreased. This resulted in negative weights of associations between stimuli that had not been presented in close succession. Even sequentially forward-facing associations developed negative weights if the temporal gap between the presentations of involved stimuli was too big. For associations to the directly following stimulus, however, the strengthening outweighed the unlearning effect.

In a situation when the weights were not trained to near their maximum values, the resulting recall was poor even for short sequences. The available amount of activation was not only spread to subsequent adjacent stimuli but also to indirectly following stimuli and back to preceding stimuli. This resulted in a flat activation distribution over the full network, i.e. after 10 cycles all units had an activation level below 0.1 and the four highest activation levels differed by less than 0.02. The activation decay caused the activation levels to decrease quickly such that after 5 cycles all activation levels were below 0.2. In order to stretch the spreading of activation over more units, the activation decay inside the units can be attenuated or the sigmoidal nature of the signal processing can be intensified until finally approaching a threshold-based way of spreading the signal. In this way already the signals of medium strength would be heavily amplified such that a smaller amount of activation is enough to highly activate subsequent stimuli.

An increased activation level in the last unit of a longer sequence could be achieved by extending the cue presentation time. But still this unit's activation level was always below 0.33 which is not considered as a strong recall. Furthermore, the recalled pattern did no longer allow to clearly conclude the temporal order in which the stimuli where perceived during training. Directly after cue release, three stimuli were highly activated at the same time with activation levels above 0.7.

In all tested scenarios on sequential material the recall of a sequence happened

faster than the original presentation. Spreading activation to a subsequent unit was done within one cycle whereas during learning each stimulus had been presented for a couple of cycles.

5.4 Comparison with Temporal Difference Learning

A popular and successful model for associative learning is the Temporal Difference (TD) model (Sutton and Barto, 1990; Balkenius and Morén, 1998). Similarly to TOSAM, the TD model can learn relationships between different inputs. Originally, TD learning was intended for classical conditioning tasks where associations are formed between conditioned and unconditioned stimuli. However, when extending this procedure over multiple stimuli, it is possible to learn an associative chain of successively presented stimuli, or an associative group when all stimuli are presented at the same time. In this regard, the TD model serves for similar purposes as TOSAM, and TD learning provides a possible alternative to the learning rule used in TOSAM. In order to compare both models, the TOSAM model was altered such that it uses the TD learning rule. In other words, the TD learning rule has been extended to work with numerous inputs in an associative network. This changed model will be referred to as the Temporal Difference Associative Memory (TDAM). TDAM works in the same way as TOSAM but changes needed to be made for the units, the associations and the processing steps within a cycle.

A unit in TDAM contains the following variables: a *data pattern* which encodes a certain stimulus, an *input* value (originally denoted as X) which indicates whether the respective stimulus is present or not, a *trace* value which models the decaying trace of a stimulus with the decay parameter δ, the aggregated *signal sum* (originally denoted as $\bar{V}$) which models the discounted prediction for this stimulus, the *old signal sum* of the previous time step, and the *prediction error*.

An association in TDAM is described by the two units it connects (*source unit* and *destination unit*) and the *strength* of the association (originally denoted as V).

Just as in TOSAM, the network in TDAM can grow dynamically. Thus, if unknown stimuli are perceived, new units are created and associations to all other units are created. As long as a certain stimulus is present, the corresponding input value is set to 1.0. A processing cycle of TDAM consists of the following steps:

1. Each unit spreads a signal to all connected units. The signal sum of a unit *dst* is computed as follows:

$$signalSum_{dst} = \sum_{i \in incoming_{dst}} strength_i \cdot input_{src(i)} \tag{5.1}$$

where $incoming_{dst}$ is the set of all associations linking to the unit dst and $input_{src(i)}$ denotes the input value of the source unit of an association i.

2. For all units the prediction error is calculated. The error of a unit dst depends on its input value, its signal sum and its signal sum of the previous time step:

$$error_{dst} = \alpha \cdot \beta \cdot (input_{dst} + \gamma \cdot signalSum_{dst} - signalSumOld_{dst}) \quad (5.2)$$

3. Associations are learned by updating corresponding strength values. The change in strength of an association i is the product of the error in the destination unit and the trace of the source unit:

$$strength_i = strength_i + error_{dst(i)} \cdot trace_{src(i)} \quad (5.3)$$

4. The traces of all units decay. For a unit src the updated trace value is:

$$trace_{src} = trace_{src} + \delta \cdot (input_{src} - trace_{src}) \quad (5.4)$$

5. Finally, the signal sums need to be stored to be available in the next time step. For a unit dst the current signal sum becomes the old signal sum:

$$signalSumOld_{dst} = signalSum_{dst} \quad (5.5)$$

Having this model, it was possible to run the evaluations described in this section again with TDAM. The parameters of TDAM were set to $\alpha = 0.1$, $\beta = 1.0$ and $\gamma = 0.95$ (Sutton and Barto, 1990). The input value $input_{src(i)}$ was set to 1 when the corresponding stimulus was presented and to 0 otherwise. The signal sum of a unit was used as an indicator for the strength of recalling the stored information.

5.4.1 Symmetric Associations

When learning a group of stimuli for 100 cycles TDAM evolved all weight values in the network equally strong. But the number of stimuli influenced the final weight value. For lower numbers of stimuli the prediction error in each unit was higher, which led to higher changes of the weight values. If the network contained more connected units, the signal sum that each unit received was higher. In turn, the error was lower and learning progressed more slowly. Just before the end of the learning phase the connection weights were ca. 6.8 for a group of 3 stimuli, and ca. 2.2 for a group of 10 stimuli. However, due to the error correction term in the learning rule, the release of the input caused a final drop in weight. Thus, the final weight values ranged from ca. 5.6 for a group of 3 stimuli and ca. 0.2 for a group of 10 stimuli.

When recalling for 10 cycles with a 1 cue stimulus, the signal sum in all non-cue units was below the trained association weights for all group sizes. For a group of 3 stimuli the signal sum in each non-cue unit was ca. 5.4; for a group of 10 stimuli the corresponding signal sum was ca. 0.2. Figure 5.10 shows the maximum signal sum for group sizes from 3 to 10 stimuli. While smaller groups could be considered as strongly recalled, groups of more than 8 stimuli were recalled only weakly.

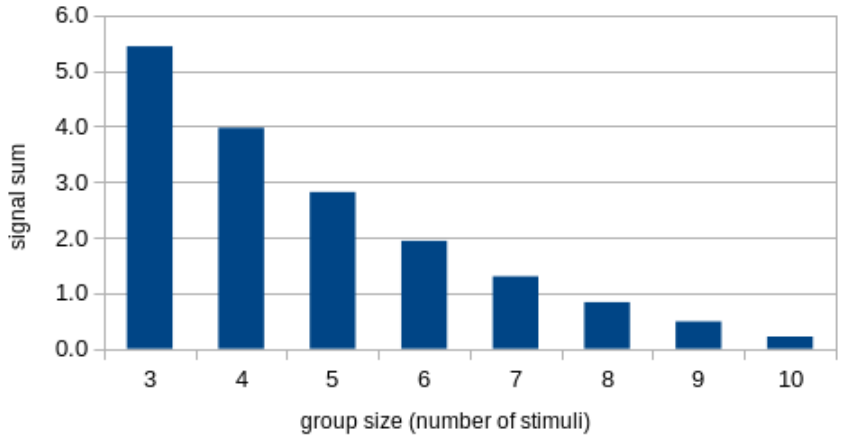


Figure 5.10: The maximum signal sum in non-cue units achieved for fully associated groups from 3 to 10 stimuli with 1 stimulus presented as a cue for 10 cycles.

In TDAM the signal sum does not accumulate over time. Thus, the maximum values were measured directly after the cue presentation. The cue unit itself had a signal sum of 0 during the whole process. Although the associations from non-cue units back to the cue unit were trained, no signals were spread over these associations. In TDAM the signal spread over a connection depends on the association strength and the input value of the source unit but not on the signal sum of this unit. Thus, signal spreading originated only from the cue unit but not from any other units in the network.

The absence of signal accumulation over time became obvious when using longer presentation times for the cue stimulus. For a group of 10 stimuli and a cue presentation time of 10 cycles, the measured signal sum was ca. 0.22. When presenting the cue for 100 cycles, the maximum signal sum in each non-cue unit was ca. 0.1. Corresponding values for all tested cue presentation times are shown in Figure 5.11. Increasing the cue presentation time resulted in a lower maximum signal sum and a weaker recall. The reason for this effect is the unlearning of association strengths during the recall process, caused by the prediction error in the destination unit. As long as the recall process goes on, the corresponding weight value will decrease.

Another option for an improved recall performance was to increase the number of cue stimuli. Figure 5.12 shows the recall performance for a group of 10 stimuli and various numbers of them used as recall cues. The maximum signal sum increased

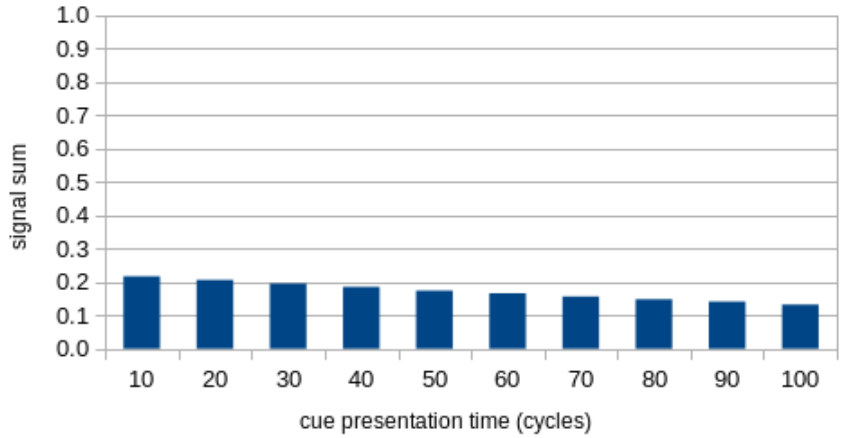


Figure 5.11: The maximum signal sum in non-cue units achieved for a fully associated group of 10 stimuli with 1 stimulus presented as a cue for different numbers of cycles.

with more cue stimuli being shown. With 9 cue stimuli the measured signal sum was ca. 1.3. Having more units with an input value of 1 led to a higher total amount of signal being spread to all other units. In turn, the signal sum in each unit was higher. If at least 2 stimuli were shown as a cue, also these cue units received a signal. Corresponding signal sum values were even higher than in non-cue units. Again the prediction error caused learning during the recall phase. This resulted in an increased strength of cue-to-cue associations and a decreased strength of cue-to-non-cue associations. When all 10 stimuli were presented as recall cues, the measured signal sum was much higher than in the case of 9 cue stimuli. In both cases, the unit in question received the signals from all 9 other units. However, the presentation of all stimuli (i.e. 10 stimuli) as a cue led to an increase in corresponding association strengths whereas, in the other case (with 9 cue stimuli), the strength values of cue-to-non-cue associations decreased during recall.

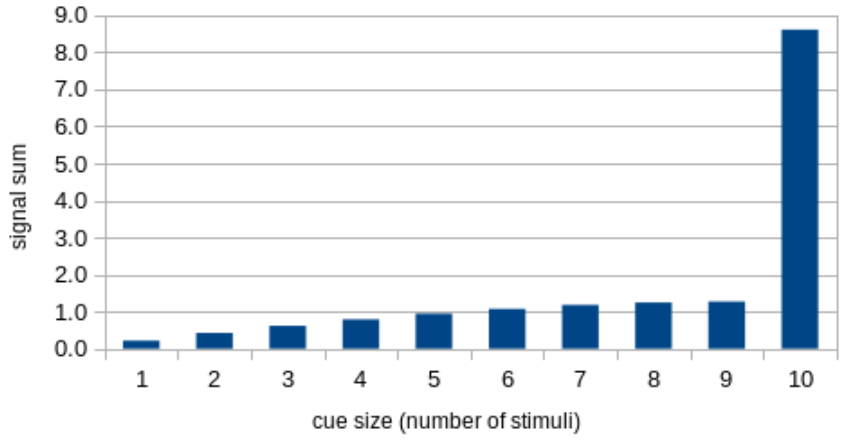


Figure 5.12: The maximum signal sum in non-cue units achieved for a fully associated group of 10 stimuli with different numbers of stimuli presented as a cue for 10 cycles.

5.4.2 Asymmetric Associations

Asymmetric associations could be successfully learned with TDAM. When presenting a sequence of stimuli in close temporal succession, associations to subsequent stimuli had positive strength values. Associations to the direct successors had the highest strength values in the network. Associations to non-adjacent stimuli were trained more weakly the further the stimuli were spaced apart in the sequence. Association strengths to sequential predecessors remained at 0. TDAM learned sequences of different lengths and each stimulus was presented for 10 cycles, followed by a gap of 2 cycles. Each sequence was repeated 25 times. After training, the first stimulus was shown as a cue for 1 cycle.

The sequential recall of the stimuli in their correct order could not be accomplished by TDAM. The spreading process did not continue along the chain of associations. In TDAM only a unit with an active stimulus generates a signal that spreads to associated units. For shorter sequences associations were trained to the last sequence stimulus such that a signal could spread to the corresponding unit. Thus, the unit of the cue directly propagated a signal to all other units including the unit of the last stimulus. In this sense, all non-cue stimuli were recalled at the same time. Figure 5.13 shows the signal sum values measured in the respective last unit for sequences of different length. While the signal sums in the second and the third units of a sequence were still reasonably high, later units only received a very small signal amount and cannot be considered as recalled.

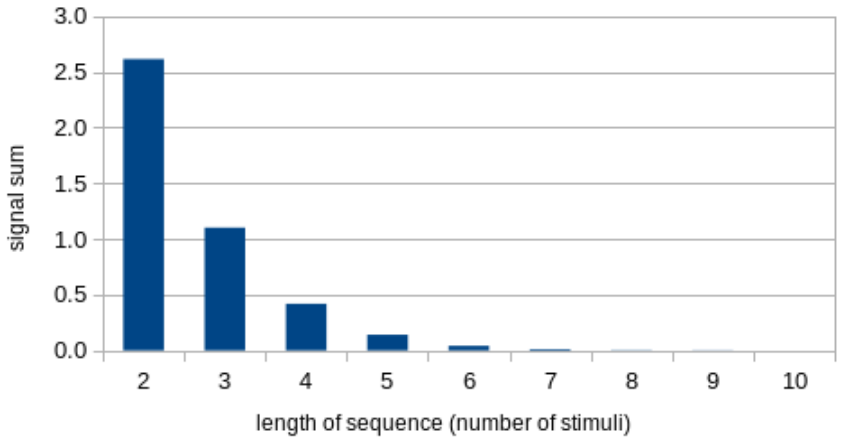


Figure 5.13: The maximum signal sum measured in the last unit of sequences of different numbers of stimuli when learning each sequence 25 times with a gap of 2 cycles after each stimulus presentation and presenting the first stimulus of the corresponding sequence as a cue for 1 cycle.

The same evaluation was performed with a modified stimulus presentation during learning – each stimulus was presented for only 10 cycles without any gap between adjacent stimuli and each sequence was repeated 10 times. Here the associations were trained similarly but corresponding strength values were higher than in the

previous run. Due to the shorter gap, the trace of each input was higher when the next stimulus was presented. This resulted in stronger associations to the direct successors. However, associations to non-adjacent stimuli were trained more weakly compared to a presentation with gaps of 2 cycles.

In accordance with the strength values, a higher signal sum was measured in the second sequence unit, but lower values in later units. The signal sum values for sequences of up to 10 stimuli are shown in Figure 5.14. Again, longer sequences could only be recalled partially and later units did not receive any signal. For sequences of 9 or more stimuli, no signal was received by the respective last unit.

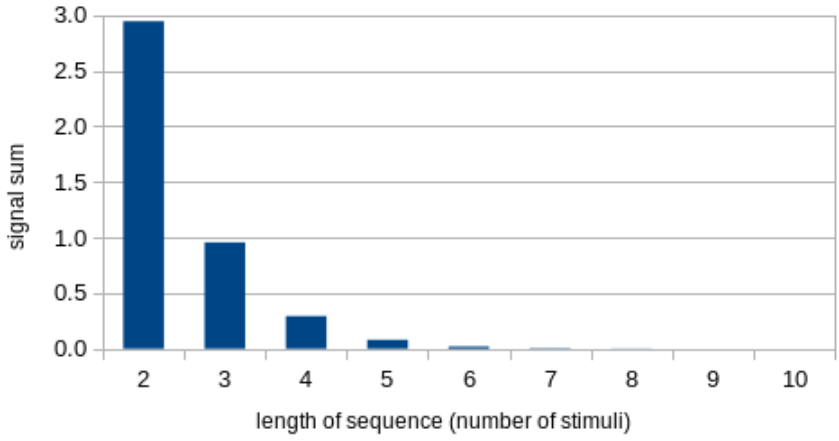


Figure 5.14: The maximum signal sum measured in the last unit of sequences of different numbers of stimuli when learning each sequence 10 times without any gap after each stimulus presentation and presenting the first stimulus of the corresponding sequence as a cue for 1 cycle.

Also by showing the cue stimulus for a longer period TDAM was not able to even weakly recall the last stimulus in a sequence of 10 stimuli. The association from the first to the tenth sequence unit was not trained at all and no signal was propagated over this association. A longer cue presentation time would not lead to a higher signal sum as there is no build-up of the signal in TDAM. Thus, the signal sum in non-cue units does not increase over time. Instead, any positive strength values of corresponding associations even decrease because of the prediction error that occurs in the last sequence unit.

5.5 Summary

Experiments on forming associations between different stimuli showed that TOSAM could easily store associations for any number of stimuli but had problems when recalling larger groups. This problem could be overcome by increasing the number of stimuli used as a recall cue, or by presenting the cue for a longer time. In both cases, more activation became available in the network, which led to a stronger recall of the

whole group of associated stimuli. TOSAM could strongly recall longer sequences if the corresponding associations were trained fully and clearly. Otherwise, the recall cue needed to be shown for a longer period, which improved the recall performance but resulted in the simultaneous recall of multiple stimuli together rather than one after the other. The model fulfils the tasks of basic associative learning but under certain conditions the recall performance is not as could be desired.

The inclusion of Temporal Difference learning did not lead to better recall results. TDAM performed worse than TOSAM in the tasks of recalling groups as well as for sequences of stimuli. TDAM learned the associations for smaller groups of stimuli successfully but did not evolve them strongly for bigger groups. Accordingly, the recall was weaker for bigger groups. When learning a sequence, TDAM could learn the temporal relationships very clearly and no associations were learned to any sequential predecessor stimuli. But the sequence could not be recalled in its correct order. Instead, TDAM recalled all stimuli at once and, for longer sequences, only the beginning of the sequence could be recalled. A major drawback of TDAM is the lack of consecutive signal propagation over multiple units. TOSAM offers this functionality and can recall sequentially trained stimuli in their correct temporal order.

Chapter 6

ICALA applied to Robot Learning

This chapter describes two experiments that were carried out to validate the functionality of the entire architecture. The first experiment (Section 6.1) targets the recognition performance of M-SOINN as well as the associative learning and recall capabilities of ICALA including TOSAM (Keysermann and Vargas, 2015). The second experiment (Section 6.2) focuses on ICALA's ability to learn incrementally and investigates how the architecture deals with larger amounts of data over a longer time period (Keysermann, 2014).

6.1 Robot Experiment: Signs and Poses

In order to evaluate ICALA in a real scenario, an experiment involving a NAO T14 robot (Figure 6.1) has been carried out. The task was to learn associations between visual patterns captured by the robot's camera and poses of the robot's arms.

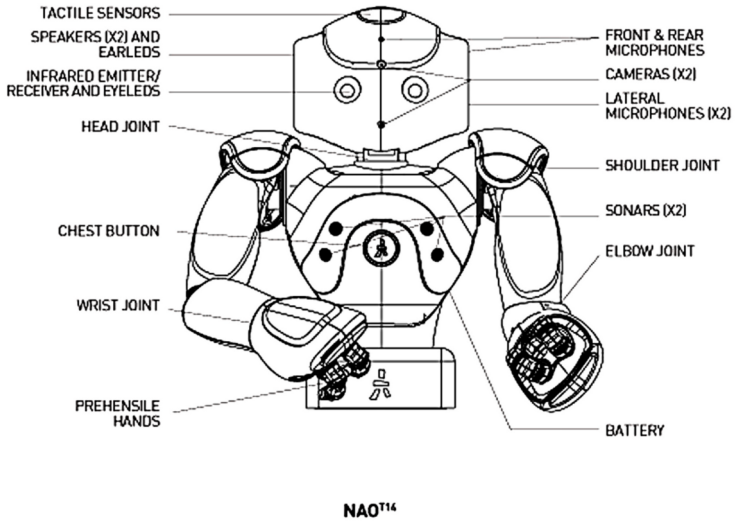


Figure 6.1: Schematic illustration of the NAO T14 robot by Aldebaran Robotics. The front camera is located centred in the upper half of the robot's head, the two arms have shoulder, elbow and wrist joints (image source: Aldebaran Robotics).

To learn an association the robot's arms needed to be held in the desired position and a visual pattern had to be placed in front of the robot's camera. The robot perceived its own arm configuration and the captured image. This configuration was kept for a certain amount of time, which led to the creation of an association between the respective inputs. Afterwards, upon being shown a previously learned visual pattern the robot had to recall the associated arm pose and move its arms into the corresponding position. Different road signs served as visual patterns[1]. During presentation the static road sign patterns were fixated on a pinboard at a fixed distance in front of the robot. However, the exact position of the sign on the pinboard varied slightly between the presentations because the experimenter attached the signs without using any markers. The experiment setup is shown in Figure 6.2.

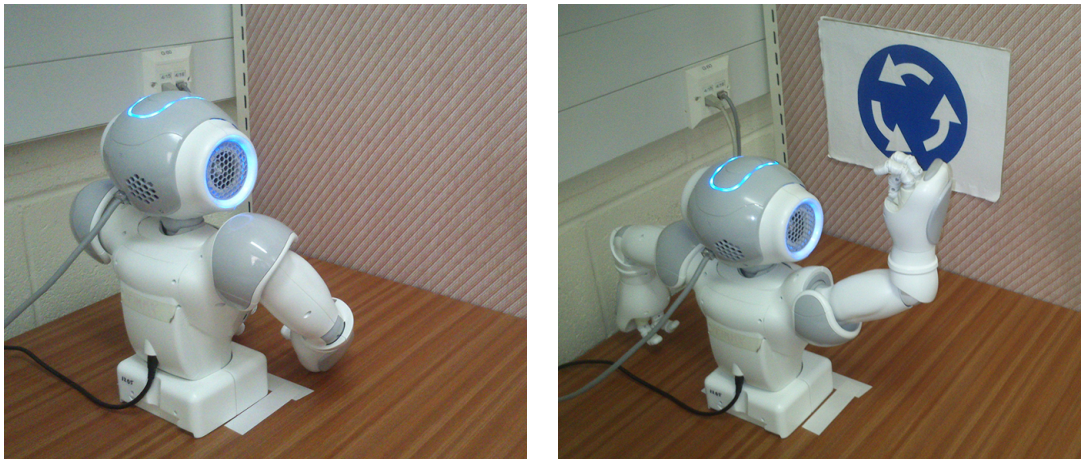


Figure 6.2: Experiment setup with the NAO robot when no sign is shown (left) and when a sign is shown and the robot takes the corresponding arm pose (right).

6.1.1 ICALA Configuration

The employed ICALA configuration comprised TOSAM and two M-SOINN modules, one for processing the visual patterns and one for handling input and output of the arm joints. TOSAM was running at a frequency of 10 Hz, which means that every 100 ms the cluster inputs from the two M-SOINN modules were processed and learning took place. For this experiment, the learning rate η in TOSAM (Equation 3.8) was set to 0.01.

6.1.2 M-SOINN Module for Visual Patterns

For obtaining a visual pattern the robot took an image from its camera at a resolution of 160 by 120 pixels. Automatic adjustments for exposure, gain and white balance were disabled. For each pixel the red, green and blue components were taken (RGB format) and the image was further resized to 80 by 60 pixels. Before being processed

[1]The road sign patterns are available at `https://github.com/MatthiasKeysermann/ICALA`.

by the respective M-SOINN module, each image was blurred with a Gaussian filter (radius = 4, $\sigma = 0.75$). Moreover, each value was normalised to be in the range of $[0, 1]$. No further treatment was applied to the pixel values. Thus, the input vector for the M-SOINN algorithm was 14400-dimensional and consisted of normalised red, green and blue values for each pixel of the image.

In order to choose the parameter values of the M-SOINN module, an initial calibration phase was performed. During this phase different parameter configurations were tested for short periods of learning (up to one minute) and the topology development was observed. The following criteria were taken into account. The algorithm should form clusters of different perceptions of the same road sign without joining patterns of other signs. The cluster formation should happen after perceiving a road sign for only a few inputs (i.e. less than 10 perceptions) and eventually lead to stable clusters containing 10 or more nodes when more inputs have been been perceived (i.e. more than 100 perceptions). Smaller clusters with 2 or 3 nodes should be kept such that they can be joined with clusters of the same road sign or incorporate more nodes when similar patterns are perceived later on. Shortly perceived inputs (e.g. when switching signs) should not be kept, in particular isolated nodes should not unncessarily inflate the topology. Accordingly, the module was set up as described in the following.

The module captured and processed a new input every 200 ms. The minimum number of nodes required for activating a unit in TOSAM was set to 2. The M-SOINN algorithm was configured in the following way. Instead of using a dynamic similarity threshold, a fixed threshold of 0.1 per dimension ($\tau = 0.1$) was used. Newly created nodes could be connected to existing nodes using the refined connection criterion. In order to remove edges, they had to have an age greater than 1000 ($age_{dead} = 1000$). The regular clean-up process happened every 30 inputs ($\lambda = 30$) during which the local error was reduced, small clusters were pruned and clusters could be joined. The parameters for removing nodes with two or one neighbours were $c_2 = 0.02$ and $c_1 = 0.01$, respectively. The decision about joining two clusters was based on a fixed distance threshold of 0.1 per dimension ($\phi = 0.1$). Other modifications were not used during the clean-up process, i.e. the longest edge was not removed, neither was the node with the minimum number of signals.

6.1.3 M-SOINN Module for Arm Poses

The arm configuration of the robot was given by the current angles of the robot's arm joints. These were the angles of shoulder roll, shoulder pitch, elbow roll and elbow yaw. The values of the wrist joint were not used. With the help of the NAOqi API[2] the angle values were read for both left and right arm, which resulted in an 8-dimensional input vector for the corresponding module. In this case, these values

[2] `http://doc.aldebaran.com/1-14/naoqi/motion/almotion-api.html`

were not normalised; the module directly used the returned angles in radians. The two arms were not treated independently by the algorithm, i.e. an entire pose was considered as a single input.

The corresponding M-SOINN module should form clusters for the inputs of each arm pose when the robot's arm are held in a certain position. Clusters should emerge within a short learning time after at most 15 inputs and become stable clusters with at least 10 nodes when more than 50 inputs have been perceived. Each cluster should not be inflated by incorporating too many inputs and contain no more than 50 nodes. Rapid arm movement (e.g. when switching poses) should not result in clusters that store in between positions. Corresponding isolated nodes should be removed from the topology. If these in between poses were stored, slow arm movement could lead to the gradual extension of the cluster of the previous pose and finally result in a single cluster for two distinct poses. For choosing the module's parameters according to these criteria, an initial calibration phase was performed where different parameter settings were tested. Then the module was set up as described in the following.

For this module the interval for reading a new input was 400 ms and the cluster size threshold for activating a TOSAM unit was set to 2. In the M-SOINN algorithm the similarity threshold was fixed at a value of 0.02 per dimension ($\tau = 0.02$) and edges with an age greater than 1000 were removed ($age_{dead} = 1000$). Not active was the modified connection criterion for new nodes. The topology clean-up was performed every 30 inputs ($\lambda = 30$). The algorithm inserted a new node to reduce the local error, the longest edge in the topology was removed and clusters were joined if the distance between their closest nodes was at most 0.02 per dimension ($\phi = 0.02$). Nodes with only two or one neighbours were removed based on the parameters $c_2 = 0.001$ and $c_1 = 0.1$, respectively. The measure to remove the node with the minimum number of signals was not applied.

6.1.4 Head Movement & Noise

The robot constantly moved its head in small steps around a centred point of view. Both the head yaw and pitch could take values between ca. -6 and $+6$ degrees (normally-distributed around 0 and limited within the specified range). This guaranteed enough variance in the input patterns of the camera, which, in turn, facilitated the formation of clusters with more than two nodes for each road sign. In this way, ICALA could generalise over similar inputs (i.e. perceived variations of each road sign), resulting in an easier recall of each particular pattern. For the same reason a small amount of noise was added to the readings of the joint angles, more precisely, normally-distributed noise around 0 with a standard deviation of 0.02 (ca. 1.1 degrees). Noise has been shown to be beneficial for other clustering algorithms such as *Expectation-Maximization* and *k-Means*, as well as for competitive learning algorithms (Osoba and Kosko, 2013). In particular, noise can speed up convergence

and seems to be especially useful in unsupervised competitive learning scenarios (Osoba and Kosko, 2013).

6.1.5 Evaluation Procedure

In this experiment the robot was learning the arm poses by demonstration. Kinaesthetic teaching (Amor et al., 2009; Akgun et al., 2012) was used, which means that the experimenter manually moved the robot's arms into the position to be learned. For practical reasons, which will be explained in Section 6.1.8, both learning and recall processes could not happen at the same time, which was originally intended. The M-SOINN module for the arm poses (Section 6.1.3) had to be configured to explicitly distinguish between learning and recall phases. During the learning phase this module was set up to only process the joint angle readings. During recall no readings were processed but the robot moved its arms to the desired position, which had been calculated based on the cluster centroids and the output obtained from TOSAM. The resulting 8-dimensional vector was used to set the desired joint angles. With the help of the NAOqi API the robot then moved its arms at a constant speed until the intended arm pose had been reached.

To evaluate learning and recall, four pairs of visual patterns and arm poses were tested (Figure 6.3). This allowed to determine how ICALA would handle and associate data obtained from different sensors. Different road signs were used as visual patterns. These were complemented by a background pattern that was associated with a robot pose of having both arms lowered.

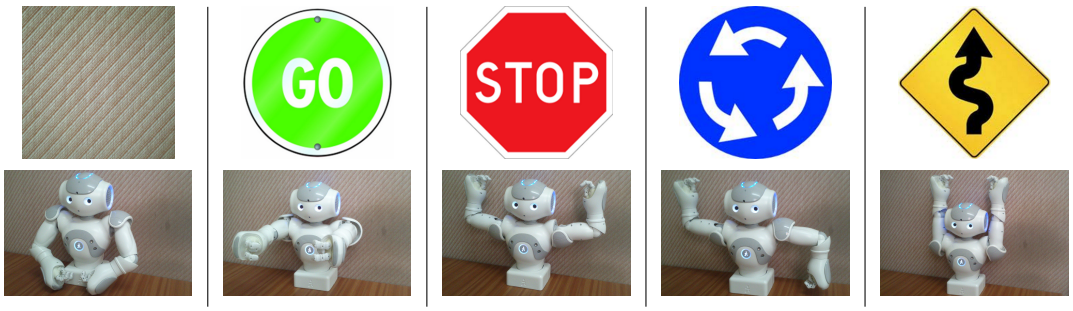


Figure 6.3: Signs and poses to be learned. Each sign had to be associated with the corresponding pose of the same column. The first column shows the background pattern (when no sign is shown) which corresponds to a pose of having both arms lowered.

The experiment was conducted in two parts. In the first part the robot did not learn associations but each M-SOINN module was tested separately on the respective inputs. This gave an impression of how fast clusters for the various input patterns can be formed and how many nodes are required to represent these inputs. In the second part the associative learning capabilities of ICALA were tested which included the recall of arm poses. To measure the progress achieved in associative learning a previously associated visual pattern was presented in front of the camera whereupon

the robot automatically positioned its arms depending on the associations with the learned arm poses. Additionally, evaluations were performed with pairs of visually similar patterns. For each sign another sign of the same colour was chosen but with a slightly different geometrical shape (Figure 6.4). This helped to judge whether the recall process heavily relied on colour information alone instead of the visual features of the entire pattern. Furthermore, the robot was tested with rotated versions of the road signs (Figure 6.4).

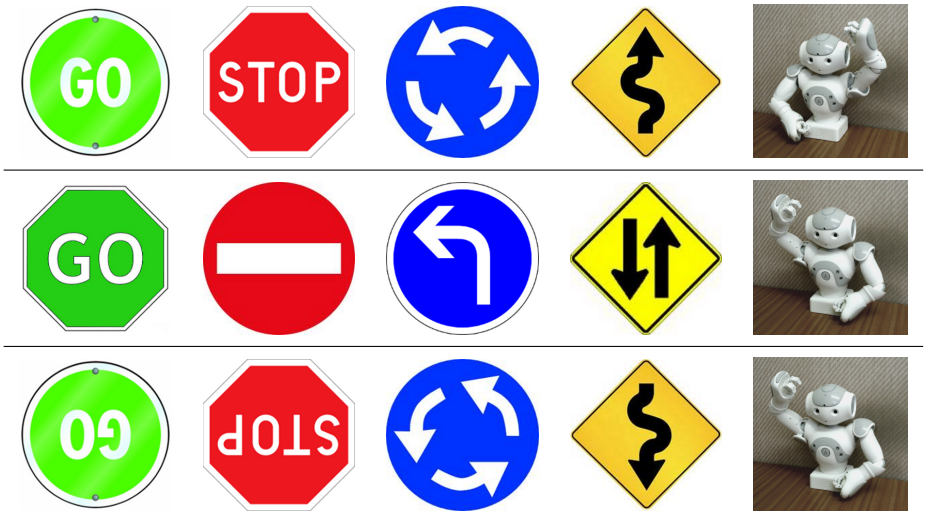


Figure 6.4: Visually similar road signs and rotated versions. The original road signs (first row) were associated with an arm pose where the left arm was raised, visually similar signs (second row) as well as the rotated versions (third row) were associated with an arm pose where the right arm was raised.

The outcomes are reported separately for the two different parts of the experiment. Section 6.1.6 describes the results of learning the input patterns, namely the different road signs and arm poses. Section 6.1.7 reports about the recall process which involves recognising the previously learned visual patterns.

6.1.6 Results: Pattern Learning

This part of the experiment investigated how long it would take to learn the various inputs and which amount of nodes are required to represent them in the respective M-SOINN module. A pattern was considered as learned when, while presenting the pattern, there was no change in the cluster structure for 60 seconds. In other words, the number of clusters had to remain constant for at least one minute. This time corresponds to $60s \cdot \frac{5}{s} = 300$ inputs of road sign patterns and $60s \cdot \frac{2.5}{s} = 150$ inputs of arm joint angles. Only clusters with at least 2 nodes were taken into account as only these clusters would cause an activation of a unit in TOSAM and would be relevant for further associative learning, i.e. only for these clusters units are created in TOSAM. To obtain more accurate estimates of the learning time and the number of nodes, this process was repeated five times for each road sign and each arm pose.

Results are listed in Table 6.1 and Table 6.2. The reported time specifies the end of the 60-seconds period without a change in the cluster structure. This means the desired clusters emerged already one minute earlier, at which time they also contained a lower number of nodes. In every case, the cluster structure remained stable only when all inputs had already collapsed into one single cluster.

								
	time (s)	nodes	time (s)	nodes	time (s)	nodes	time (s)	nodes
	250	79	415	107	250	82	340	85
	415	105	315	99	360	106	225	65
	510	120	370	111	465	129	215	61
	285	92	385	107	400	115	195	62
	335	91	655	150	475	122	215	59
μ	359.0	97.4	428.0	114.8	390.0	110.8	238.0	66.4
σ	104.7	15.6	132.0	20.2	91.4	18.2	58.1	10.6

Table 6.1: Learning results for different road signs.

Table 6.1 shows that the average time required for learning stable representations for each of the four signs ranges from ca. 4 minutes to ca. 7 minutes. With a cycle time of 200 ms these durations correspond to around 1200 inputs and 2100 inputs respectively. This amount of inputs was compressed by M-SOINN into a smaller amount, which made up the corresponding cluster. The average number of nodes required for representing each of these signs in a single cluster ranges from ca. 66 to 115. The exact values differ from sign to sign but in general a longer learning time resulted in a higher number of nodes inside the corresponding cluster.

The background pattern is not included in Table 6.1. Learning this pattern happened much faster than learning the four road signs. In less than 10 seconds a cluster emerged which then consumed all additional inputs. The background pattern was never represented with more than one cluster. After ca. 1 minute this single cluster still contained less than 10 nodes.

As can be seen in Table 6.2, each of the four poses were learned in a shorter time compared to the road signs. In none of the tested cases more than 2 minutes were required to form a stable representation of a pose. As the corresponding M-SOINN module was running at a frequency of 2.5 Hz, in a 120-seconds time frame, 300 inputs were perceived. Considering that the obtained times include a period of 60 seconds without any change in the number of clusters, an arm pose can be learned in less than a minute. The amount of nodes used to store an arm pose in the topology averages at roughly 25 nodes with only little variance in the actual results.

Learning the pose with both arms lowered was not tested here. However, very similar results as for the four tested poses can be expected because the input is as

								
	time (s)	nodes	time (s)	nodes	time (s)	nodes	time (s)	nodes
	91	29	83	24	107	29	95	28
	82	23	83	27	83	21	83	20
	106	25	68	24	68	26	80	22
	118	22	108	31	94	25	75	21
	106	28	118	24	104	26	94	29
μ	100.6	25.4	91.0	26.0	91.2	25.4	85.4	24.0
σ	14.1	3.0	20.4	3.1	16.0	2.9	8.8	4.2

Table 6.2: Learning results for different arm poses.

well an 8-dimensional real-valued vector. Although the actual values are different, the amount of variance in the inputs (in terms of noise) is the same. The learned cluster is only placed in another region of the input space.

6.1.7 Results: Associative Learning and Recall

The second part of the experiment consisted of the robot perceiving a road sign and the corresponding arm pose at the same time in order to learn an association in TOSAM. After having learned this association for a certain amount of time, the road sign was removed and the robot lowered its arms while seeing the background. The association with the background pattern had been trained prior to learning of any other association. Then the road sign was shown again and the robot needed to recall the associated arm pose. This recall process included recognising the sign by assigning the visual perception to an existing cluster. The recognition activated the corresponding unit in TOSAM and caused activation spreading to the associated arm pose. An output vector was calculated by the arm poses module and the robot positioned its arms in the recalled position. If the robot achieved the desired arm pose within 10 seconds (100 cycles in TOSAM), the recall was rated as a success.

The results of part 1 suggest that a learning time of around 5 minutes should lead to a high recognition rate for at least three of the road signs. Since associations can already be developed as soon as units are created, the recall performance for associated arm poses (that can be learned in a shorter time) can be expected to be good for such a duration of learning. It was decided to directly choose a shorter time for learning in order to explore whether good recall can already be obtained with shorter learning times. At first, a learning time of 3 minutes was chosen, i.e. the robot perceived each pair of road signs and arm poses for a period of 180 seconds. During this period, ICALA received 900 visual inputs and 450 arm joint readings. For each of the four sign-pose pairs the described process of learning and recall was

repeated five times. In every run, M-SOINN could form a single cluster for each road sign as well as a single cluster for each arm pose within the 3-minutes learning period. The associations between each pair were strongly trained with an associative strength of more than 0.9. For all signs the learned arm pose could be recalled immediately when the sign was shown, resulting in a recall rate of 100%.

For a second series of runs, the learning time was decreased to only 1 minute (corresponding to 300 visual inputs and 150 arm joint readings). Again, the process was repeated five times for each association. Still an overall recall rate of 100% was achieved. The robot was always able to recall the associated arm poses for all signs. However, the emerged cluster structures for the road signs were different for this shorter learning time. A sign was no longer represented by one single cluster but by multiple clusters with different perceptions of a sign. An example of such a cluster structure is shown in Figure 6.5. The shorter training time did not allow M-SOINN to join all these smaller clusters into one bigger cluster. Nevertheless, for each cluster an association with the respective arm pose was learned in TOSAM. These associations were trained more weakly with associative strengths of less than 0.3. But as there were no competing associations (e.g. with different arm poses), still a successful recall was achieved within 10 seconds. Recalling the associated arm pose did not always happen immediately. Sometimes the perceived visual pattern was not recognised as a familiar pattern, i.e. the M-SOINN algorithm formed a new cluster for which no associations existed. But after a few seconds, this new cluster was joined with an existing cluster which itself had already been associated with the desired arm pose. Also due to the head movement of the robot, the image captured by the camera changed constantly and within 10 seconds (50 visual inputs) the captured image changed to a pattern that was represented by an existing cluster. As a result, the road sign was recognised and the associated arm pose could be recalled.

Additionally, the recall performance was evaluated with pairs of visually similar road signs. If colour information played a predominant role in the recall process, the robot would not be able to distinguish between the respective two visually similar signs. To test this assumption it was necessary that the robot associated each two signs with two different arm poses. Thus, one sign was associated with a pose where the left arm was raised, and the other sign with a pose where the right arm was raised. As before, the architecture was tested with learning times of 3 minutes and 1 minute. After each learning phase the robot was shown both signs of the previously learned pair, one after the other, and had 10 seconds (100 cycles in TOSAM) per sign to recall the associated arm pose and position its arms accordingly. Again, five runs per pair of signs were performed.

In all runs the robot could always recall the correct pose, resulting in a recall rate of 100%. M-SOINN never grouped two visually similar signs into a single cluster. Instead, the corresponding topologies contained separate clusters for each two distinct

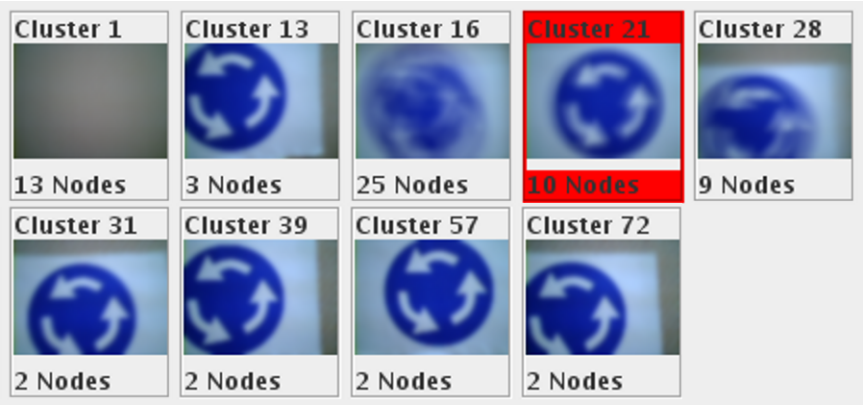


Figure 6.5: Example cluster structure for learning a road sign for 1 minute. Each rectangle shows a different cluster. Only clusters with at least 2 nodes are shown. Displayed for each cluster are the identifier, the cluster mean and the number of contained nodes. Cluster 1 represents the background pattern, all other clusters represent different perceptions of one road sign. Not all variations have been joined into one single cluster yet. Highlighted in red is the cluster of the current input.

patterns, e.g. both when learning for 1 minute and when learning for 3 minutes. Due to the longer duration, the associations with the correct pose were stronger for a learning time of 3 minutes. But also when learning for only 1 minute the robot perfectly recalled the correct pose.

Furthermore, the same recall task as above was executed, but instead of using similar signs the robot was shown the four original signs rotated by 180 degrees. Recalling both poses separately was possible only for the blue "roundabout" sign. For all other signs the robot could not distinguish between corresponding perceptions. The rotated pattern ended up in the same cluster as the original pattern. This made it impossible for the robot to correctly recall two different arm poses. Instead both poses received activation when showing either of the signs (as both signs were recognised as the same sign). The result was an arm pose with both arms partially raised. In two cases the secondly learned pose predominated during the recall process and the robot raised its right arm slightly higher than its left arm.

6.1.8 Analysis

The reported results of the first part (Section 6.1.6) show that the arm poses were much faster to learn than the road signs. Moreover, less nodes were required to create stable representations for the arm poses compared to the road signs. In this regard, the dimensionality of the input data plays an important role. A vector of 8 different real values represented an arm pose whereas a visual pattern required 14400 values. A well-known phenomenon in machine learning is the *curse of dimensionality* that states that generalisation becomes harder as the dimensionality of the inputs increases (Domingos, 2012). This effect is clearly reflected in the obtained results.

Furthermore, all captured inputs of one visual pattern had a lot more variance

than only the perceived joint angles. This was due to the head movement of the robot, which moved the capture area nearly perpendicular to the direction the camera was facing. This movement process added translation invariance into the cluster representation of a visual pattern and allowed for better recognition performance.

The process of forming stable clusters took several minutes, which may appear rather long for learning a single pattern. However, all the reported durations are dependent on the set cycle times. By setting shorter cycle times the reported learning times would decrease. With a cycle time of 200 ms M-SOINN can process 300 inputs per minute. Forming a stable cluster of a road sign took approximately 5 minutes, which corresponds to 1500 inputs. With a cycle time of 10 ms the same amount of data would take only 15 seconds to learn. An arm configuration could be learned within 3 seconds by using a cycle time of 20 ms for the corresponding M-SOINN module.

As revealed in the second part of the experiment (Section 6.1.7), the robot was able to perfectly recall the associated arm poses. This was the case in all tested scenarios although the strengths of corresponding associations were different for learning times of 3 minutes compared to learning periods of only 1 minute. The reason for this equally good recall is the fact that activation in TOSAM can build up over several cycles. While a road sign is shown and recognised, the corresponding unit in TOSAM gets activated, i.e. raises its activation level to the maximum value and spreads activation to other units. The actual amount of activation to be spread depends on the strength of the corresponding association and is less in each cycle for smaller strength values. But the amount of activation can build up over several cycles as long as a cue (a road sign) is shown.

Although a representation of one cluster per visual pattern is elegant, there is no need for actually learning the topology up to that point. The recall scenario with a 1-minute learning time showed that also a scattered topology structure could lead to a very good recall performance. In this case, one visual pattern was represented by several clusters and each cluster contained different perceptions of this pattern. Associations were trained for all these clusters, which allowed proper recall independently of which exact perception was received as a cue. Hence, representing a pattern with multiple clusters is sufficient. As long as the learned clusters cover enough variability, exhaustive cluster joining is not required. Nonetheless, a single cluster is more desirable for TOSAM because less units and associations need to be stored and processed.

The additional evaluation with visually similar signs showed that the architecture does not solely (or even mainly) rely on colour information when recognising visual patterns. Signs of the same colour but with a different geometrical shape could be correctly distinguished. The representations in M-SOINN were still general enough to tolerate some translation variance. Although the recognition of geometrical features

is not trivial (Husbands et al., 1998; Vargas et al., 2014), M-SOINN could recognise all different signs in the given setup. However, for three road signs M-SOINN did not distinguish from the rotated versions. The perceived patterns were too similar and eventually perceptions of the rotated and the non-rotated version were clustered together. The differences between these patterns were very small and M-SOINN generalised over them. This generalisation strength can be adjusted with parameters of M-SOINN, namely the fixed similarity threshold and the absolute join tolerance for clusters. If desired, smaller values can be chosen and M-SOINN will be more sensitive to small details. The intermediate pose of having both arms partially raised was caused by the averaging of the cluster centroids to produce the output vector. Both learned poses had roughly an equal amount of activation and the output arm joint angles were between the originally learned values. In the cases when the right arm was raised slightly higher, the unlearning process had weakened associations with the first pose while learning the second pose. An association with the first pose was already trained. But when perceiving the rotated version as the same visual pattern, the expected arm pose input (i.e. first pose) was not received, which led to a decrease in weight of the corresponding connection.

The similarity thresholds in the M-SOINN algorithm were set to fixed values for both modules. In general, fixing threshold values is not optimal because it pre-sets the coarseness of the topology. This way, however, the algorithm becomes insensitive to small differences which means that similar inputs more likely form one cluster rather than several smaller clusters. Pre-setting similarity thresholds works well when the type of data and, in particular, the desired cluster outcome is (at least roughly) known in advance. More precisely, the fixed threshold values determine at which granularity level the algorithm should separate inputs into different clusters. Normally, the similarity thresholds should be determined by the data itself. By using the dynamic adaptation method that is included in M-SOINN, the algorithm is able to both distinguish between fine-grained differences in the data and form very specific small clusters, but also bigger coarse clusters with more variance in the contained inputs. However, if very similar inputs are received consecutively (instead of interchangeably with highly different ones) the similarity thresholds of corresponding nodes quickly become very small. This results in many small clusters with each containing only a few nodes. Consequently, the formation of clusters that represent a whole pattern category (e.g. a specific road sign) becomes hard in such a topology.

Also absolute values were set for the distance tolerance thresholds used for the cluster joining process. In this way the decision to join clusters is no longer dependent on the average node distance within each cluster. If this average node distance is very small for certain clusters, a joining is not possible even if the corresponding clusters are positioned very close to each other in the topology. An absolute tolerance value

allows to join such clusters and, thus, facilitates a faster formation of bigger clusters.

A drawback in the experiment was that the robot needed to be manually indicated when to learn and when to recall an arm pose. In general, such a restriction can be considered as undesirable for autonomous robots that should operate without any further intervention of the experimenter. An autonomous robot should learn from new inputs while recalling stored knowledge at the same time. For adopting such a setup in the conducted experiment, the robot would have to constantly recall an arm pose and move its arms accordingly. For the robot to learn a new association, the experimenter had to manually move and hold the robot's arms in the desired position while the corresponding visual pattern was shown. To do so while the robot is recalling an arm pose, the experimenter would have to apply force against the servomotors' torque, at least as long as the robot tries to adopt a different pose. Hence, this approach would only be possible with low torque applied to the servomotors. But in this case the torque would not be strong enough for the robot to lift and position its arms during recall. A higher torque, on the other hand, prevents having the arms moved manually by the experimenter. Due to these practical limitations the robot only applied torque to its arm joint motors when manually set to do so during the recall phase.

6.2 Robot Experiment: Faces and Labels

In another experiment the incremental learning capabilities of ICALA were tested. Accordingly, the amount of information to be learned and memorised should increase gradually. Throughout the whole learning process the acquired knowledge must be ready to be recalled. Hence, ICALA had to retrieve all inputs perceived so far in regular intervals.

A dataset for such a task should be reasonably large but also allow the architecture to generalise over similar inputs. The AT&T database of faces[3] was used, which contains portrait images of 40 different people. For each of these persons 10 images exist which show different variations of facial expression, head rotation, etc. In total the dataset consists of 400 different images. A single image is 92 by 112 pixels in size and stores greyscale values.

6.2.1 Evaluation Procedure

The steps executed during the experiment were the following. At first, only one person of the dataset was learned for a certain time; all 10 different images for this person were repeatedly presented to the system in a random order. At the same time a text label served as another input which contained the text "Person1". This ensured that the presented images were associated with the label. Thereafter, each

[3] `http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html`

image presented so far was shown once again. This time, however, the label was not given as input but needed to be recalled. During this recall phase, a blank (entirely black) image was shown in between any successive portrait images. Then another learning phase was carried out for the next person in the dataset together with the label "Person2". The following recall phase included all images seen so far, presented in a random permuted order. This procedure was continued by introducing one person after the other until the architecture had seen the images of all 40 people. After each recall phase the recognition rate was determined, i.e. for how many images the correct person was identified. The exact time spent for learning each person was 240 seconds (4 minutes); and during this phase a single face was presented for 4 seconds. During the recall phase each portrait was shown for 12 seconds, as was the blank image.

Investigated were two different scenarios: one was a simulated scenario where each image was directly fed into the architecture; the other one involved a NAO T14 robot[4]. In the latter, the images were magnified and displayed on a large 22 inches screen which was observed by the robot with its front camera (Figure 6.6). Moreover, the recalled label was spoken out by the NAO's built-in text-to-speech system. The following sections list specific parameter settings for each of these setups.

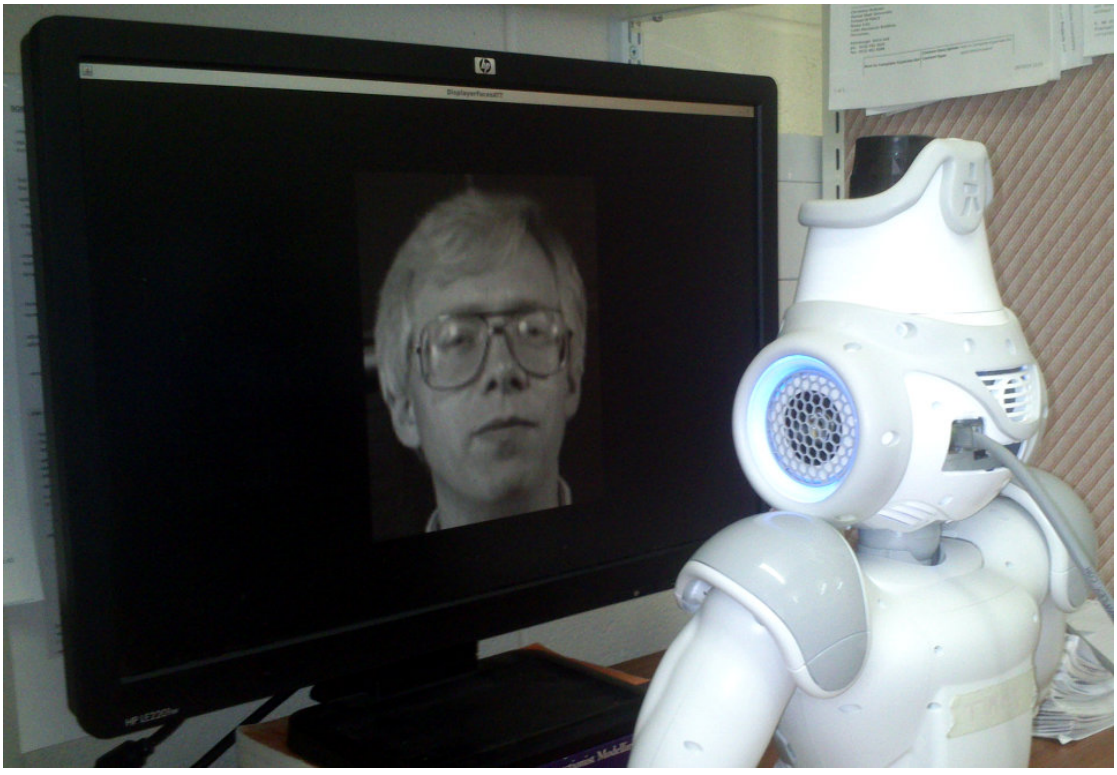


Figure 6.6: The NAO robot observes the image of a person's face on a screen.

[4] `http://www.aldebaran.com/en/humanoid-robot/nao-robot`

6.2.2 Simulated Scenario

In the simulated scenario ICALA included TOSAM and two M-SOINN modules: one for clustering the face images (in the following referred to as the *FacesATT* module) and one for storing the text labels (referred to as the *TextLabel* module).

Before running the experiment, different parameter configurations were tried by testing only a subset of the data (with up to ten persons). In this way, configurations could be avoided which would not allow to form clusters of each person's face variations or falsely cluster together the faces from different people. The *FacesATT* module should be able to store the faces of all 40 people. Dynamic similarity thresholds allow to create clusters of two nodes when receiving novel perceptions, such that novel face variations would directly result in a new cluster. Such clusters should be joined into a bigger cluster containing more variations of a single person's face. Cluster joining should happen independent of the average node distance within each cluster. Thus, an absolute join tolerance was used. As this can lead to falsely joined clusters, the longest edge in the topology should be removed in each clean-up step to recover from such situations. Additionally, the reduction of the local error should stabilise the topology. The *TextLabel* module should store the numerically represented label data, such that each label is represented by a single cluster with 2 nodes. These small clusters should not be pruned at all. The exact configurations of both modules are described in the following.

The *FacesATT* module was running at a cycle time of 2000 ms. While learning the images of one person, the M-SOINN algorithm received 120 inputs during these 4 minutes. The module passed on cluster inputs to TOSAM for clusters containing two or more nodes. No artificial noise was added to the images. As the images were processed in their original size, the input vector had 10304 dimensions. Greyscale values were normalised into the range $[0; 1]$. Dynamic similarity thresholds were used but the modified connection criterion for new nodes was disabled. The maximum age of edges was $age_{dead} = 50$ and a clean-up step was performed every $\lambda = 50$ inputs. Nodes with only one or two neighbours were removed based on the parameters $c_2 = 0.001$ and $c_1 = 0.01$. Enabled were the modifications for removing the longest edge and for reducing the local error. Also cluster joining was possible with an absolute join tolerance of $\phi = 0.1$ per dimension. Table 6.3 lists all parameter settings for this module.

The cycle time set for the *TextLabel* module was 2000 ms and the threshold for giving cluster inputs to TOSAM was 2. Similarity thresholds were dynamically calculated. Other parameters were set to $age_{dead} = 1000$, $\lambda = 100$, $c_2 = 0.0$ and $c_1 = 0.0$. None of the modifications were active for this module. An incremental numbering scheme was used to map the labels to numeric values.

TOSAM was running at 1 Hz, which corresponds to a cycle time of 1000 ms. As each recall phase of a person's face lasted 12 seconds, TOSAM had 12 cycles to

input dimensions	10304		cycle time	2000 ms
value range	[0,1]		cluster activation threshold	2
fixed similarity threshold	-		connect new nodes	no
remove minimum-density node	no		reduce local error	yes
λ	50		c_1	0.01
age_{dead}	50		c_2	0.001
remove longest edge	yes		join clusters	yes
relative join tolerance	-		absolute join tolerance	0.1

Table 6.3: Parameters of the M-SOINN module *FacesATT*.

spread enough activation to the associated text label unit.

Furthermore, another setup was tested with some changes in TOSAM. One change was to not limit the amount of activation a unit can spread. Instead of dividing the available amount of activation, each unit should be able to spread more activation than available – the actual amount spread should depend only on the respective connection weight and the attraction of the receiving unit. Another tested option was to reduce the number of outgoing associations for each unit by creating connections only between units with a positive input load. The intention here was to reduce the amount of activation spread over irrelevant associations, in particular over connections whose weights would become negative due to unlearning. Further tested was a variant where the unlearning part in the Hebbian learning rule was disabled, i.e. $\chi = 0$ such that $contr_v = (l_v)^\rho$ with $\rho = 15$ (Equation 3.6). With this change no inhibitory associations could develop and positively trained associations did not decrease their weight values during recall (apart from the very weak weight decay).

Additionally, ICALA has been evaluated with minor changes to the learning procedure. The time for learning a person has been doubled, such that each person was learned for a total duration of 8 minutes. In this way, the associations could develop stronger weights and the effect of the unlearning process during recall would be mitigated. All other experiment parameters were left unchanged and TOSAM was used without any variants.

6.2.3 Robot Scenario

In the robot scenario ICALA included a module for processing the input read from the NAO's camera (referred to as *NAOCamera* module), a module for providing and speaking out the labels (referred to as *NAOTextToSpeech* module), as well as TOSAM for creating corresponding associations.

Apart from the noise being inherent when reading an image from the camera's sensor, some transformations were applied to the images on the screen to bring the whole setup closer to situations where actual people would show up in front of the robot's camera. As a person would never place their head in exactly the

same position, a translation between 0 and +/-10 pixels randomly displaced the image along both the x and y axes. For similar reasons, the images on the screen were randomly scaled by a factor between -1% and +1%. Furthermore, each image was randomly rotated, either clockwise or counter-clockwise, by an angle of up to 1 degrees.

The M-SOINN modules were calibrated by testing different parameter configurations with a subset of the data (with up to ten persons). In this scenario, the *NAOCamera* module had to deal with additional transformations of the images and noise captured by the photo sensor. Due to the transformations applied to the images, a specific perpection may be seen only once during learning but still needs to be incorporated into a cluster. Thus, the formation of clusters was supported by allowing M-SOINN to connect new nodes. Using fixed similarity thresholds allows to generalise over the amount of noise in the inputs. For the same reason, an absolute join tolerance was used for joining clusters. Falsely joined clusters should be split apart again by removing the longest edge in the topology. Cluster growth was supported by inserting another node into the topology when reducing the local error. The *NAOTextToSpeech* module should perform in the same way as the *TextLabel* module and store the numerically represented label data. Each label should be represented by a single cluster with 2 nodes and these clusters should not be pruned. Corresponding parameter choices for both modules are described in the following.

The *NAOCamera* module captured a greyscale image from NAO's front camera at a resolution of 160x120 pixels and rescaled it to 80x60 pixels which resulted in a 4800-dimensional input vector. Greyscale values were normalised into the range $[0; 1]$. The cycle time of the module was 2000 ms. Hence, the module received 120 inputs when learning a single person for 4 minutes. A cluster needed to have at least two nodes to create activation in TOSAM. The similarity threshold of the M-SOINN algorithm was set to a fixed value of $\tau = 0.07$ per dimension. The modified criterion for connecting new nodes was active. Edges had to have an age of $age_{dead} = 50$ or greater to be removed. Every $\lambda = 50$ inputs a clean-up step was executed. The parameters c_2 and c_1 were set to $c_2 = 0.001$ and $c_1 = 0.01$. The modifications for removing the longest edge, for reducing the local error and for joining clusters were active, with an absolute join tolerance of $\phi = 0.07$ per dimension. Table 6.4 lists all parameter settings for this module.

The cycle time for the *NAOTextToSpeech* module was 2000 ms; the threshold for a cluster to create activation in TOSAM was 2. No additional noise was generated and the algorithm calculated dynamic similarity thresholds. No modifications were used in this module and other parameter settings were $age_{dead} = 1000$, $\lambda = 100$, $c_2 = 0.0$ and $c_1 = 0.0$. An incremental numbering scheme was used to map the labels to numeric values.

The cycle time set for TOSAM was 1000 ms. During recall each person's face

input dimensions	4800		cycle time	2000 ms
value range	[0,1]		cluster activation threshold	2
fixed similarity threshold	0.07		connect new nodes	yes
remove minimum-density node	no		reduce local error	yes
λ	50		c_1	0.01
age_{dead}	50		c_2	0.001
remove longest edge	yes		join clusters	yes
relative join tolerance	-		absolute join tolerance	0.07

Table 6.4: Parameters of the M-SOINN module *NAOCamera*.

was shown for 12 seconds; this corresponds to 12 cycles within TOSAM.

Further tested was another setup, which was identical to the previously described robot scenario but used stronger transformation for the displayed images. This should represent a more realistic situation where the position of a person's head in front of the camera would vary more widely. Here, the displacement of each image was between 0 and +/-20 pixels along each axis, the scale factor was between +/-10%, and the maximum value used for rotating the images was 5 degrees.

Also in the robot scenario increased learning times were tested. For these evaluations the duration for learning a single person was set to 8 minutes. No changes were made to the recall procedure or to any other experiment settings. The transformations applied to the images on the screen were the same as described initially, i.e. no stronger transformations were used.

The following sections present the results of both scenarios. They will look into how the topology for the people's faces developed but will primarily focus on the recall rate.

6.2.4 Results: Simulated Scenario

In the simulated scenario ICALA could incrementally learn and memorise associations for up to 37 people. For the remaining three persons neither associations existed in TOSAM nor clusters were created in both M-SOINN modules. At this point the *FacesATT* module had to process a topology with 2045 nodes, 1927 edges and 118 clusters. The underlying hardware was no longer capable to process the amount of data within the set cycle times. Inputs of the remaining persons were missed and no clusters were formed for these people. Yet, TOSAM and the *TextLabel* module were still responsive under a higher load as the recorded recall rates were still above 0. Despite the high amount of nodes and edges, the number of clusters generated in the *FacesATT* module remained below 120 while still preserving most face variations of each person (Figure 6.7). The corresponding topology contained always more clusters than persons but the cluster count always stayed far below the total number of presented faces. On average the algorithm created ca. 3 clusters per person

($\frac{118}{37} \approx 3.2$ clusters/person).

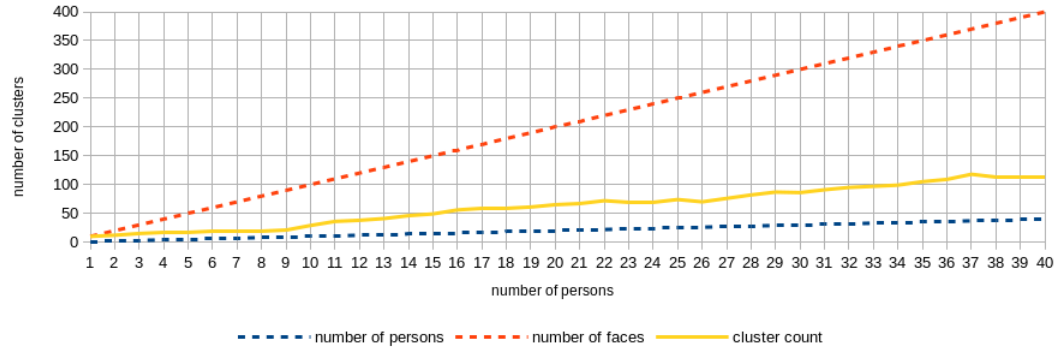


Figure 6.7: Simulated Scenario: Topology development for the *FacesATT* module.

For up to 36 persons, nevertheless, the recall performance of the architecture can be investigated. Figure 6.8 shows the ratio of correctly identified faces while the architecture was incrementally learning more people. Overall, the rate of recall started high and gradually decreased as more people had to be recalled. For the first two persons the recall was perfect. Then a temporary drop to 0.4 happened when 4 different persons had been learned. An examination of the topology of the faces revealed that M-SOINN clustered inputs from person 3 together with inputs from person 4. Thus, at least for 10 out of 40 faces TOSAM recalled the wrong person. Accordingly, the recall rate is ca. 25% lower than the rates for 3 and for 5 persons. For 6 to 13 people ICALA correctly identified more than 70% of the memorised faces. The recall rate finally reached a value of around 0.2 after having seen 36 people.

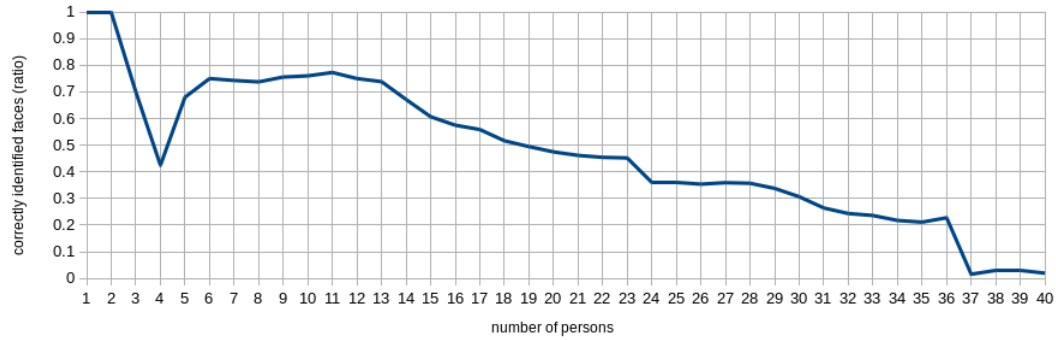


Figure 6.8: Simulated Scenario: The ratio of correctly identified faces plotted against the number of persons learned incrementally.

The actual counts of correctly identified faces are shown in Figure 6.9. While the number of correctly identified faces constantly increased for up to 13 people, from then onwards ICALA did not recognise any more faces. The count of correctly identified faces remained roughly the same until 36 people had been learned. At this point, 82 faces were correctly identified. For 37 people the number of recognised faces dropped to 6 faces. The *FacesATT* module was not responsive anymore and did not process most of the presented faces. Only for the 6 recognised faces, the

input activated the corresponding cluster (and unit in TOSAM) and the correct label could be recalled.

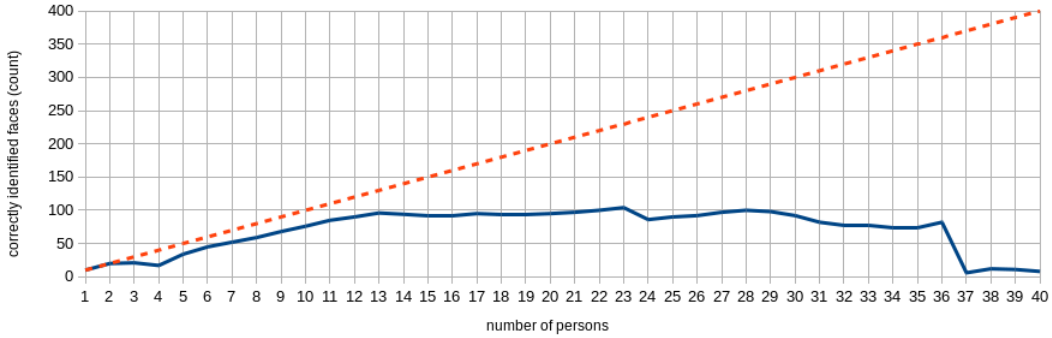


Figure 6.9: Simulated Scenario: The number of correctly identified faces plotted against the number of persons learned incrementally. The red line shows the total number of faces presented so far.

6.2.5 Results: Simulated Scenario (TOSAM Variants)

When using ICALA with the TOSAM variants, the architecture could learn associations for all 40 people. Although no changes were made to the M-SOINN algorithm, the obtained topologies in the *FacesATT* module contained less than 800 nodes and less than 700 edges. The numbers of clusters were roughly the same as in the simulated scenario without variants (Figure 6.10). With the option to allow overspreading the final topology of the *FacesATT* module contained 790 nodes, 672 edges and 118 clusters. When the TOSAM network was not fully connected, the *FacesATT* module stored 751 nodes, 638 edges and 113 clusters. Without unlearning the associations in TOSAM, the number of clusters in the *FacesATT* module was higher with a value of 123 but only 730 nodes and 607 edges were stored. With all variations the cluster count temporarily dropped for 33 people but otherwise increased steadily. This drop was caused by a topology refinement step which removed 21, 26 and 35 clusters for the variants with overspreading, reduced connectivity and without unlearning, respectively.

Without the unlearning part in the TOSAM learning rule the recall performance was worse than before with the recall rate being mostly between 20% and 40% (Figure 6.11). Also with the option to allow an overspreading of activation, no better recall performance could be obtained (Figure 6.11). Already after having learned 5 persons the recall rate dropped below 20%. After having seen the faces of 32 people, the recall rate improved but remained below 40%. Compared to these variants, with reduced connectivity in TOSAM the recall rate was higher in 25 (out of 40) recall phases (Figure 6.11). For lower numbers of people (from 3 to 10) the recall rate was below 0.3. The recall rate for 10 people was 0.22, whereas a recall rate of 0.94 was achieved for 11 people. For 10 people, 39 units existed in TOSAM, i.e. on average 3.9

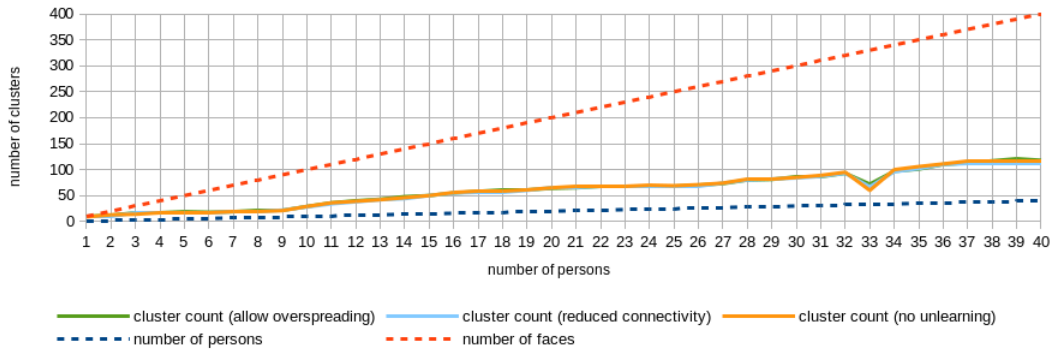


Figure 6.10: Simulated Scenario (TOSAM Variants): Topology development for the *FacesATT* module.

units per person. TOSAM with reduced connectivity creates only the associations between the faces of a person and the corresponding label. Assocations to other units are not created and cannot develop negative weights. Thus, all available activation is spread between the units of one person and the corresponding label. Once activated, these units attract a smaller amount of activation (due to a lower signal attraction). As a result, the available activation is spread within this group of units for a longer time, which is further extended due to the sigmoidal signal processing. The smaller the group, the smaller the total amount of activation decay, and the longer the activation can be maintained. If one group maintains the highest amount of activation in the network for a longer time, the corresponding label can dominate during the recall. With an increase in the average number of units (as observed for 11 people) the available amount of activation can be distributed over more associations. Without one label dominating, other labels can be recalled and higher recall rates can be achieved from then on.

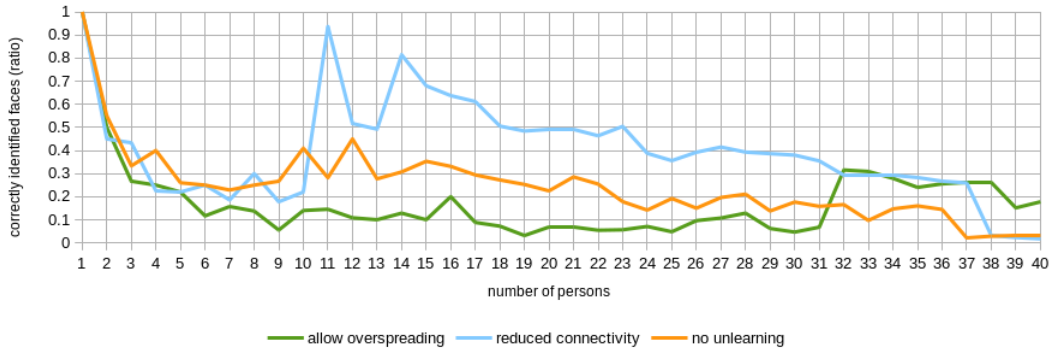


Figure 6.11: Simulated Scenario (TOSAM Variants): The ratio of correctly identified faces plotted against the number of persons learned incrementally.

6.2.6 Results: Simulated Scenario (Longer Learning Times)

Learning for a longer time could partially improve the recall performance of ICALA in the simulated scenario (Figure 6.12). With the longer learning times of 8 minutes all the recall rates for 14 or more people were higher than with learning times of 4 minutes. For 14 and 15 people, almost a perfect recall was achieved with recall rates above 0.95. Then the recall rate remained around 70%, before gradually decreasing to ca. 40%. When recalling with less than 14 people, the recall performance of ICALA was unstable with local minima at 8 people (23% recall) and at 12 people (38% recall). From 13 to 14 people, the recall rate increased from 0.39 to 0.96.

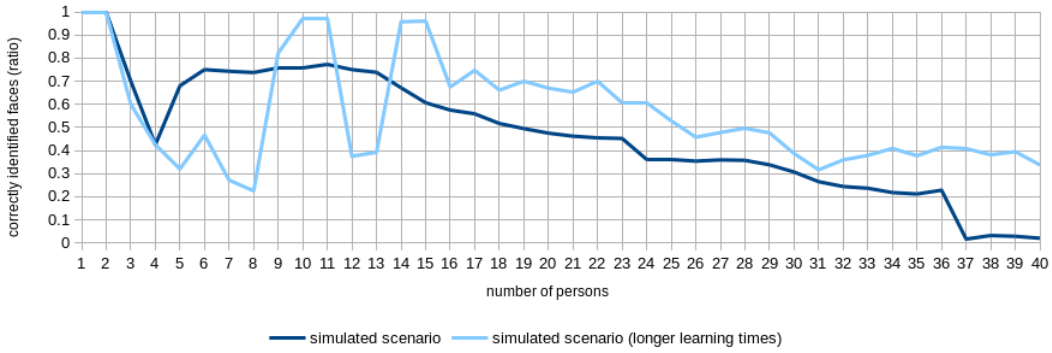


Figure 6.12: Simulated Scenario (Longer Learning Times): The ratio of correctly identified faces plotted against the number of persons learned incrementally.

The topology for 13 people shows that M-SOINN was able to form a single cluster with 16 nodes for person 13. This cluster included 90% of the respective face variations. In TOSAM, the associations between the corresponding unit and the associated text label unit were strongly trained with weights above 0.994 in both directions. In the following recall phase, whenever a face of person 13 was presented, the corresponding unit got activated and activation was spread to the associated label unit. The sigmoidal signal processing counteracted the activation decay. The label unit itself could then spread its activation back to the former unit. In this way, the two units maintained a near maximum level of activation amongst them and the label dominated when recalling with other faces. This means that, when a face of another person was presented, the activation level in the associated label unit was lower than in the previously activated (but still dominant) label unit. This resulted in a recall rate below 0.4. However, due to the unlearning of weights during recall, the mentioned connection weights decreased and had values below 0.991 after the recall phase. This was enough to allow the respective activation levels to decay, i.e. these units did no longer maintain a near maximum activation level. When learning the face variations of person 14, M-SOINN formed six clusters with at most 6 nodes each. In TOSAM, all associations to the corresponding text label unit had weight

values below 0.972. When recalling with 14 people, no single label unit dominated anymore and a recall rate above 0.9 was achieved.

6.2.7 Results: Robot Scenario

In the robot scenario the *NAOCamera* module generated clusters for up to 36 people. Although labels were created for the remaining four persons, ICALA could not learn associations for these people due to the missing clusters in the topology of the images. The maximum number of clusters the *NAOCamera* module could handle was 115. At this point, the topology contained 672 nodes and 689 edges. These two numbers are significantly lower compared to corresponding node and edge counts in the simulated scenario. However, the cluster number was roughly the same. The development of the number of clusters happened similarly as in the simulated scenario. The cluster count was always higher than the number of persons but lower than the total number of faces (Figure 6.13). The 10 face variations of a single person were represented by ca. 3 clusters on average ($\frac{115}{36} \approx 3.2$ clusters/person).

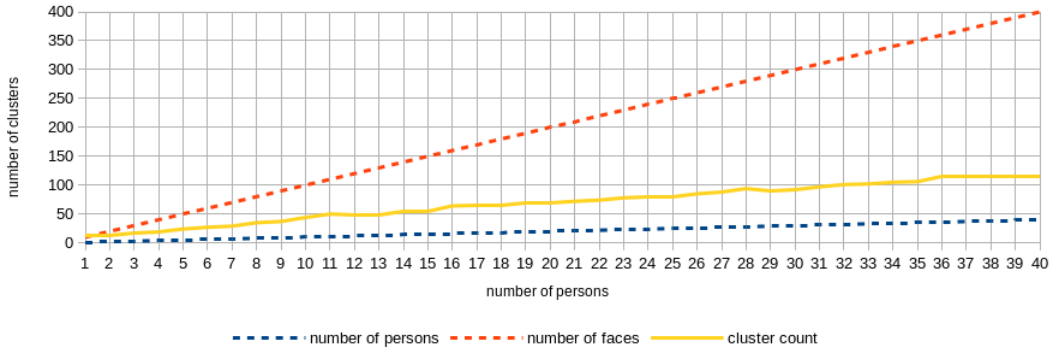


Figure 6.13: Robot Scenario: Topology development for the *NAOCamera* module.

Figure 6.14 shows the ratio of correctly identified faces. After a perfect recall for the first two persons, the recall rate decreased to nearly 31% for 9 people. When recalling 10 people, the rate increased and remained above 59% until 21 people had been presented. Another drop follows from 0.76 for 21 people to 0.07 for 22 people. Then the ratio of correctly identified faces stayed between 20% and 50% for 24 to 35 people, before the architecture reached its computational limit with 36 people. For less than 22 people, most recall rates (15 out of 21) are above 0.5. For more than 21 people, all 19 recall rates are below 0.5.

In these cases, ICALA tended to recall more recently learned labels rather than the correct labels. This is likely an effect of the network size and the number of connections with negative weights. During recall, no associations are trained. Instead unlearning happens in all connections because the associated labels are not given as inputs. This unlearning process results in connections with negative weights. The earlier a person has been learned, the more often this person has been recalled and the more decreased have the corresponding weights. Consequently, the earlier

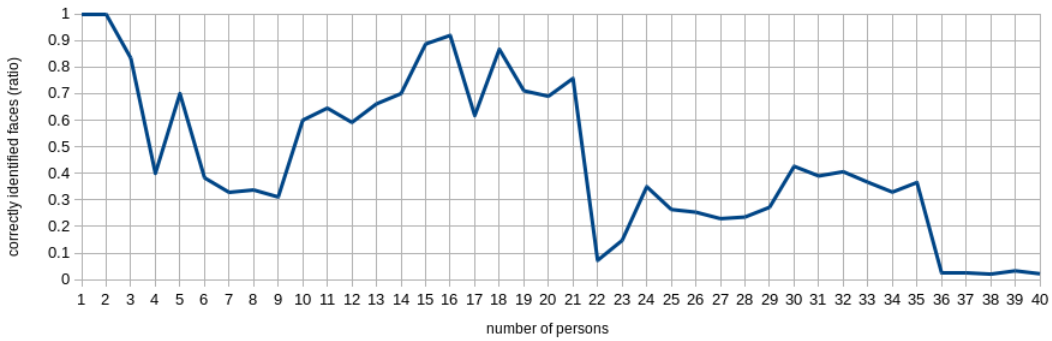


Figure 6.14: Robot Scenario: The ratio of correctly identified faces plotted against the number of persons learned incrementally.

a person has been learned, the weaker is the respective recall. Activation is not only spread over the positively-weighted connection to the correct label but also over negatively-weighted connections to other units. While learning more people, the size of the TOSAM network increases and with it the number of connections whose weights will become negative. Accordingly, the amount of available activation is distributed over more connections and the amount spread to the correct label becomes smaller. Eventually, not enough activation can be spread anymore to the correct labels, which results in a drop in the recall rate. It seems that this point was reached after 21 people.

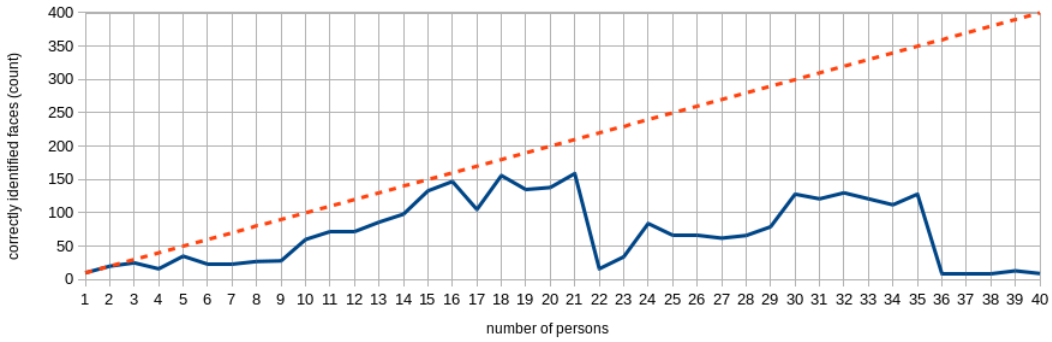


Figure 6.15: Robot Scenario: The number of correctly identified faces plotted against the number of persons learned incrementally. The red line shows the total number of faces presented so far.

Figure 6.15 shows that the number of correctly identified faces remained close to the total number of presented faces for the first 21 persons. Then the architecture suddenly fails to correctly identify even a small number of faces. As Figure 6.15 is based on the same data as Figure 6.14, the observed drop has the same cause and seems to be an effect of the network size and the number of connections with negative weights. For more people ICALA could again improve its recall performance but not recover its previous performance – less than half the number of presented faces could be identified correctly.

6.2.8 Results: Robot Scenario (Stronger Transformations)

With stronger transformations ICALA could only learn a tiny number of associations. The additional transformations applied to the faces shown on the screen led to a rapid increase of the number of clusters in the *NAOCamera* module. The corresponding topology stored 317 nodes, 201 edges and 117 different clusters after only 4 people had been presented. Figure 6.16 shows the development of the number of clusters over the whole learning period. Already for person 1, the algorithm produced 40 different clusters. The cluster count increased to 73 for two persons and to 87 for three persons. Then the number of clusters rose to 117 from where upon it remained steady. Having reached the computational limit of the underlying hardware, the module no longer processed any inputs. As a result, neither clusters were formed for the remaining people nor did the module activate any existing clusters during recall.

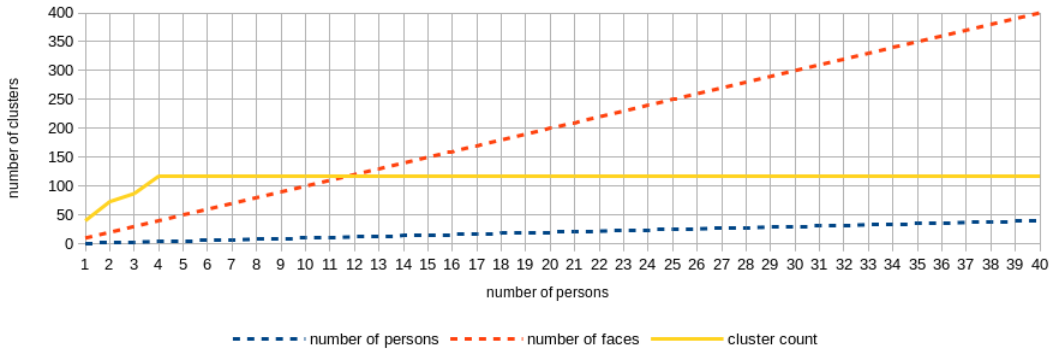


Figure 6.16: Robot Scenario (Stronger Transformations): Topology development for the *NAOCamera* module.

The early failure in processing new visual inputs is reflected in the recall performance (Figure 6.17). For a topology with two and three persons the recall rate was reasonably high. More than 70% of the learned faces were recognised correctly. While learning person 4, the *NAOCamera* module could no longer process the incoming inputs which resulted in a sudden drop of the recall rate down to 25%. The learning process of the labels was still going on and corresponding units were created. Although these units decayed their activation, the most recently learned label always had an activation level higher than all previous labels. Thus, ICALA always recalled the most recently learned label which was correct for the 10 faces of the most recently learned person. Accordingly, the recall rate is 0.2 for 5 people, 0.167 for 6 people, and converges towards 0 as more people are recalled.

6.2.9 Results: Robot Scenario (Longer Learning Times)

In the robot scenario the longer learning times (of 8 minutes per person) did not result in an improved recall performance. Like with shorter learning periods (of 4 minutes per person), the recall rate showed heavy fluctuations with similar local

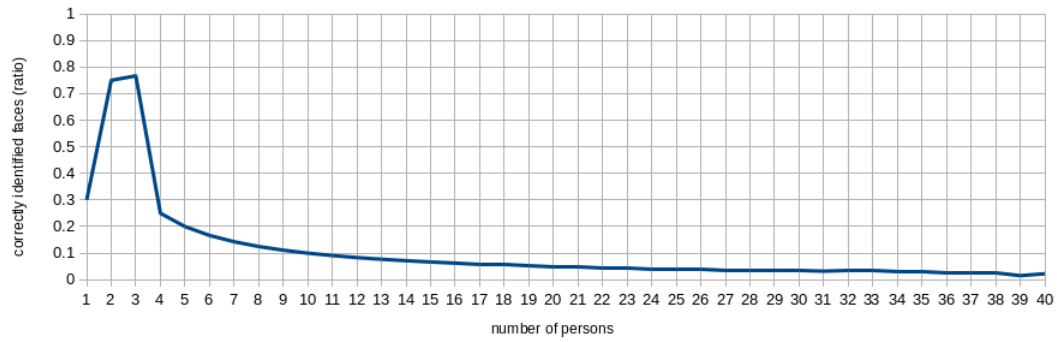


Figure 6.17: Robot Scenario (Stronger Transformations): The ratio of correctly identified faces plotted against the number of persons learned incrementally.

minima (Figure 6.18). However, compared to shorter learning times, the recall performance was worse in 25 out of 40 recall phases. Especially after heaving learned the faces of 14 people, subsequent recall rates were lower than the respective rates obtained with the shorter learning times. This poor recall performance lasted until recalling with 30 people where the recall rate suddenly increased to around 50%, but afterwards dropped below 10% again for 33 people or more.

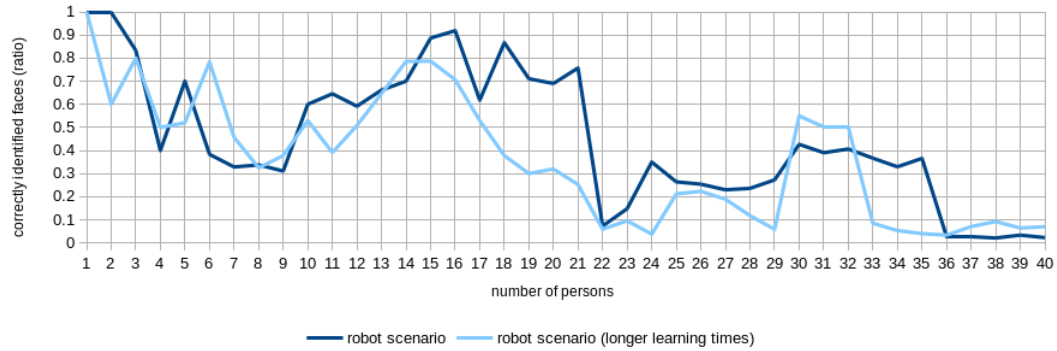


Figure 6.18: Robot Scenario (Longer Learning Times): The ratio of correctly identified faces plotted against the number of persons learned incrementally.

6.2.10 Analysis

ICALA showed limited recall performance in the given incremental learning task. Already for a small number of patterns no perfect recall was possible. For 20 people almost half of the learned faces could be identified correctly in the simulated setup. In the robot setup the architecture temporarily achieved higher recall performances but the recall rate fluctuated including sudden drops.

The poor recall performance could have originated from the clustering result of M-SOINN. If the topology did not contain all face variations of a person, clearly certain variations could not be recalled at all. Either these faces had never been stored in the topology or had been removed afterwards during a clean-up step of the algorithm. However, by looking at the actual topologies, no significant lack

in face variations could be noticed. For all people most of the face variations are represented in the topology. Figure 6.19 shows the topologies of the *FacesATT* and *NAOCamera* modules after having learned 20 people. With help of the confusion matrices it became obvious that mainly persons learned earlier (e.g. the first 7 of 20 people) could not be identified correctly. For these people, the architecture tends to predict the label of a more recently learned person. This suggests that TOSAM, in particular the unlearning of weights during recall, caused the decrease in recall rate. The more often a person has been recalled, the more decreased have corresponding connection weights. After a recall phase with 20 people, person 1 has been recalled 20 times, whereas person 20 has been recalled just 1 time. The connection weights corresponding to person 1 have been decreased 20 times more often than the connection weights corresponding to person 20. This influences the spreading process and the recall, i.e. the recall for person 1 is likely to be weaker than the recall of person 20. Figure 6.20 shows the confusion matrices for the recall with 20 people. In the simulated scenario, the faces of persons 4 to 7 were never identified correctly. In both scenarios, the wrong label was recalled for at least 5 faces of each of the persons 1 to 3.

In the simulated scenario the recognition rate decreased gradually. During recall the unlearning process of TOSAM makes the association weights more and more negative. Also positive weights decrease as there is no relearning of previously trained associations. A unit spreads activation to all other nodes but the negative weights can overweight and the major amount of activation is spread over connections with negative weights. Consequently, the amount spread over the positively-weighted association to the correct label becomes tiny.

The poor recall performance could have originated from the associative spreading process in TOSAM. If a specific face was correctly assigned to an existing cluster but the association to the corresponding label was too weak, then not enough activation could be spread to make the respective unit the most highly activated in the network. Moreover, a lot of activation was spread over negatively trained associations to other units. This amount increased over time when – due to unlearning – corresponding association strengths decreased further. As a result, the amount spread to the correct label became insignificantly small. Without unlearning, the connection weights did not decrease during recall and no negative weights could develop for connections to other units. But the achieved recall performance was worse than when including the unlearning process. The negative weights would normally suppress the activation of other units and lead to a cleaner recall of the associated information. It seems that such a competitive process is beneficial and even necessary for a better recall performance. Allowing a unit to spread more activation than available also worsened the recall performance. With too much activation in the whole network, no clear recall was possible anymore. When reducing the connectivity in the network, the

Figure 6.19: Topologies of the M-SOINN modules *FacesATT* (simulated scenario) and *NAOCamera* (robot scenario) after having learned 20 people. Each rectangle shows a different cluster. Displayed for each cluster are the identifier, the cluster mean and the number of contained nodes.

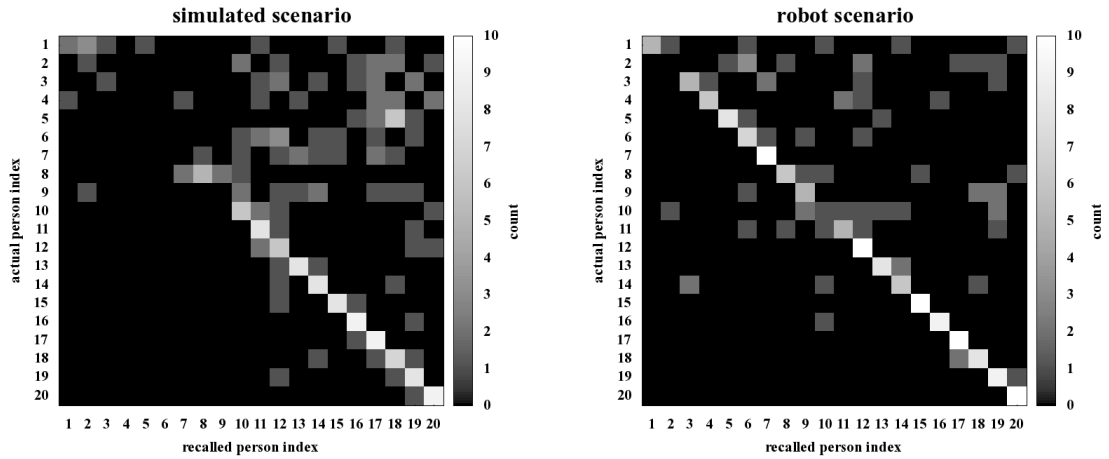


Figure 6.20: Confusion matrix for the recall with 20 people.

amount of activation spread to irrelevant units was smaller which allowed for a higher activation of the unit with the correct label. Especially for a higher number of units, the recall performance was better than with a fully-connected network. In this regard, a sparse connectivity seems to be beneficial for bigger networks with more than 50 units. This variant also has computational advantages as the algorithm works with a smaller number of connections.

Increasing the time spent for learning each person led to higher recall rates when already having learned many people. For lower numbers of people the recall was slightly unstable with sudden drops in the recall rate. Although ICALA showed a better overall recall performance with longer learning times, such a measure still does not guarantee a proper recall over extended periods of time and cannot be considered to remedy the permanently decreasing recall rate.

Also in the robot scenario the architecture could not deliver satisfying results. Although for certain numbers of people the recall rate was higher than in the simulated scenario (i.e. in 16 of 40 recall phases), the temporary drops did not allow to judge the performance as being better in the robot scenario. The recall performance fluctuated in the robot scenario whereas being fairly consistent in the simulated scenario. In the *NAOCamera* module the M-SOINN algorithm produced a lower number of nodes and edges, i.e. 672 nodes and 689 edges compared to 2045 nodes and 1927 edges in the *FacesATT* module. This was likely caused by the usage of a fixed similarity threshold. Yet, a high number of 115 clusters emerged which eventually rendered the module unresponsive and unable for continued processing.

Especially in the robot scenario with stronger transformations the architecture severely suffered from the computational constraints. The architecture could no longer keep up with storing and recognising the presented faces. For each person, M-SOINN generated a high number of clusters which quickly brought the *NAOCamera* module to its computational limits after having learned only a few people. The main difference in this scenario was the introduction of stronger image transformations.

M-SOINN was not able to combine all different transformed versions of a particular face into one cluster.

Clusters with all variations of a person's face could be achieved by increasing the tolerance value for joining clusters. But this would also make it more likely that faces of different persons end up in one cluster. The modification for removing the longest edge could only partially deal with the amount of falsely joined clusters.

Overall, the whole approach is computationally intensive. M-SOINN stores many nodes for representing several variations of a person's face. In the simulated scenario 2045 nodes were stored for 37 people, which corresponds to $\frac{2045}{37} \approx 55.3$ nodes per person and 5.53 nodes per face variation. In the robot scenario the topology contained 672 nodes for 36 people, corresponding to $\frac{672}{36} \approx 18.7$ nodes per person and 1.87 nodes per face variation. As TOSAM normally maintains a fully-connected network, already for a few units the number of connections is high. For instance, for only 10 units 90 connections are required, while a network with 100 units contains 9900 connections. To allow the underlying hardware to handle all the required computations within a single cycle, for both M-SOINN and TOSAM cycle times of at least one second were set. In accordance with that, long periods needed to be chosen as well for learning and recall. However, these restrictions are solely caused by hardware limitations. Given faster (or parallel processing) hardware, all time-related parameters could be decreased to provide a more realistic scenario.

6.3 Summary

The experiment described in Section 6.1 evaluated the pattern recognition capabilities and the associative learning performance of ICALA. Patterns could be recognised independently of small variations or noise in the input data. During learning, M-SOINN experienced small variations in the input data. These variations were necessary for the formation of stable representations. By becoming insensitive to small changes in the input, M-SOINN could reliably recognise the learned patterns. The experiment revealed that highly stable representations require an extensive amount of time to learn, which further depends on the dimensionality of the input data. Nonetheless, the results show that a good recall performance can already be achieved with much shorter learning times when the input categories are not perfectly reflected in the topological structure yet. A perfect recall was achieved in every tested case except when learning the rotated versions of the visual patterns. Here the architecture generalised over minor differences and recognised rotated and non-rotated versions as the same sign. In every other case ICALA correctly distinguished even between similar signs and the robot could successfully recall the arm pose which had been previously associated with the respective visual cue. No extra information specific to road signs was given to or engineered into the approach.

The performance of the architecture was also evaluated in an incremental learning task (Section 6.2), which involved the memorisation and grouping of portrait images of various people, as well as the association of these images to corresponding labels. Despite being able to learn nearly all presented images, the associated labels could not always be retrieved correctly. While the number of patterns and associations increased, the ratio of correctly recalled information decreased. Several possible causes for the decrease in recall rate were identified and modified versions of the architecture with corresponding countermeasures were re-evaluated. Increasing the time for learning allowed ICALA to form stronger association weights but did not result in higher recall rates in general. Extending the learning periods led to higher recall rates in the simulated scenario. But this was not the case in the robot scenario where the recall performance was worse. For TOSAM, changes in both the spreading process and in the learning rule were tested and also a variant with reduced connectivity in the associative network. The former two changes did not lead to a strictly better recall performance. However, without using a fully-connected network the recall performance could be improved as the network grew bigger and contained more than 50 units.

Chapter 7

Discussion

This chapter provides a critical analysis of various features of ICALA and presents a more in-depth discussion of relevant aspects. At first, the focus is on the architecture as a whole, including its modular structure and its way of handling the stored information. Then certain characteristics of M-SOINN and TOSAM are discussed, including specifics of each method as well as possible issues with the processing time.

7.1 ICALA

7.1.1 Modularity

The modular structure of ICALA allows the exchange of specific components independently. For instance, the TOSAM component could be replaced with TDAM (Section 5.4) or another model for associative learning. It is further possible to use a different incremental clustering algorithm such as E-SOINN for a specific modality. This allows testing for improvements with other algorithms without having to change the entire architecture.

ICALA is capable of dealing with data from multiple modalities. In regular intervals corresponding modules read sensory inputs and write outputs to actuators. The entire process is a permanently ongoing sensing-acting cycle. Every sensory state can become associated with one or more actuator states. A set of input values can be mapped to a set of output values. Thus, the architecture incrementally generates an input-output mapping between sensory and actuator states. These states include data of all connected modules, and the formed associations always involve the inputs of all modules. However, certain situations may require to focus the attention on only a few of the concurrently incoming inputs. This would require certain M-SOINN modules to stop transferring their data to TOSAM, or alternatively to influence the activation intensity for specific units in TOSAM by controlling the applied activation level. A more general suggestion for an attention mechanism will be given in Section 8.3.

7.1.2 Information Storage

ICALA reads unprocessed sensory inputs and groups them into clusters. These clusters represent abstract categories. The association of the categories of different modalities can be considered as the formation of entire concepts. In this regard, a concept of a real-world entity is made up of numerous perceptions of this entity.

According to two popular views in cognitive psychology (Frixione and Lieto, 2013), concept representation (or category learning) as done by humans can occur in the form of exemplar storage or prototype storage. The former considers a concept as the union of many single exemplars whereas the latter suggests that a prototype is derived from the perceptions of this category. Empirical evidence also exists for the usage of a combination of these two storage forms (Malt, 1989). M-SOINN employs exemplar storage while the alteration and removal of exemplars are possible. The exemplars are provided as real-valued vectors and are grouped based on their similarity. While inputs are perceived, groups (or clusters) of similar inputs emerge. Each cluster can be seen as the abstract representation of a certain input category. During the course of learning, the stored instances are altered and tend to become more similar to the average of the perceived inputs, depending on the exact connectivity. In this sense, one instance of a cluster can be seen as a prototype of the corresponding category which renders the approach as a hybrid of exemplar and prototype storage.

However, the approach of storing exemplars can quickly lead to a large amount of data that needs to be processed. Especially when dealing with high-dimensional data, M-SOINN can become a processing bottleneck (Section 7.2.1). The required computation time in a certain M-SOINN module can become too long, such that the module cannot transfer the data to TOSAM within an acceptable time frame, i.e. the actual duration of a cycle can be longer than the set cycle time. In turn, TOSAM can no longer properly correlate the incoming stimuli with the ones of other modalities if a module sends its activated cluster identifier delayed. Furthermore, with high-dimensional data another problem can occur with generalising over many similar inputs (Domingos, 2012). The higher the dimensionality of the data, the more instances are required for M-SOINN to form a cluster that covers the variance within each of the dimensions. A solution to both of these problems would be to reduce the dimensionality of the data before it is processed by M-SOINN, which will be further discussed in the following section.

7.1.3 Dimensionality Reduction

A well-known method of dimensionality reduction is the Principal Component Analysis (PCA) (Bishop, 2007). PCA first builds the covariance matrix of the data. Then the method computes the eigenvectors and eigenvalues for this matrix. The eigenvectors with the highest eigenvalues correspond to the directions with the

maximum variance in the data. Starting with the highest eigenvalues and proceeding in descending order, a certain number of the corresponding eigenvectors are then chosen to form a linear space of lower dimensionality. Finally, the original data is projected onto this lower dimensional space. Thus, PCA represents the data more compactly while preserving maximum variance (Bishop, 2007). The PCA algorithm, as described, works offline and requires the whole dataset to be known in advance. This makes the algorithm inadequate for the application with incremental online learning methods. There exist, however, some online variants of PCA. For instance, Warmuth and Kuzmin (2008) describe an algorithm whose runtime grows quadratically with the number of dimensions in the data.

If PCA is used together with M-SOINN, the same input should always be projected onto the same point in the lower dimensional space (or at least be reasonably close to it). But if the projection matrix changes over time, an issue can arise. If a later projection of a specific input differs too much from an earlier one, M-SOINN might not be able to assign the later perception to the same cluster as the earlier perception – despite both being the same inputs. This would prevent M-SOINN from correctly recognising previously encountered inputs and would severely compromise the recall capabilities of ICALA.

The dimensionality of the input data can be reduced in a more static way by using a pre-specified set of signal filters and by applying these filters consistently to all inputs. Clearly, such filters must be general enough to adequately function for all sorts of possible inputs. Possible candidates are the Haar wavelet filters which have been successfully used with audio fingerprinting (Ke et al., 2005), face detection (Viola and Jones, 2001) and object recognition (Papageorgiou et al., 1998). In the cited works, the entire datasets were used to construct an optimal set of filters obtained from Haar wavelets of different sizes and orientation. Determining such a set in advance is not possible when learning incrementally. Instead, a general set of filters must be chosen in advance. For visual image data, such a set could be a hierarchical quadtree of filters where each leaf contains four Haar wavelets – one with horizontally differing intensities, one with vertically differing intensities and two with diagonally differing intensities. Up to a certain hierarchical depth, the wavelets would be repeated over the image in smaller scales (Figure 7.1). For applying the filters to audio data, first the audio signal must be windowed and translated into the frequency domain. Successive spectra can be combined into a spectrogram, which can then be processed like an image (Ke et al., 2005).

However, a drawback of the mentioned approach is that something needs to be known about the nature and structure of the data. For instance, suppose an image is represented by a high-dimensional vector where each dimension corresponds to one pixel. Only by knowing that these dimensions have a specific arrangement, this vector can be interpreted as a planar image and a specific set of filters can be applied.

Although the structure of the data may be known for most sensors (or implied by the design of the sensor), the requirement of additional data-specific information violates the generality of any learning approach. In principle, the wavelet filters measure the difference in intensity between specific subsets of dimensions. By choosing random subsets of all dimensions, no information about the structure of the data would be required anymore. However, these filters would reflect arbitrary intensity patterns which may be unsuitable for capturing the patterns present in the incoming data.

Another challenge is to sufficiently reconstruct the original data from the reduced data. The reconstruction is required if ICALA uses the resulting data vector as an output. But the filter process leads to a loss of information and the original data cannot be reconstructed entirely – unless a sufficiently high number of filters is used. But the number of filters required for a lossless compression, i.e. to enable the exact reconstruction of the original data, would exceed the dimensionality of the data. Clearly, the total number of filters should be lower than the dimensionality of the data in order to achieve reduction.

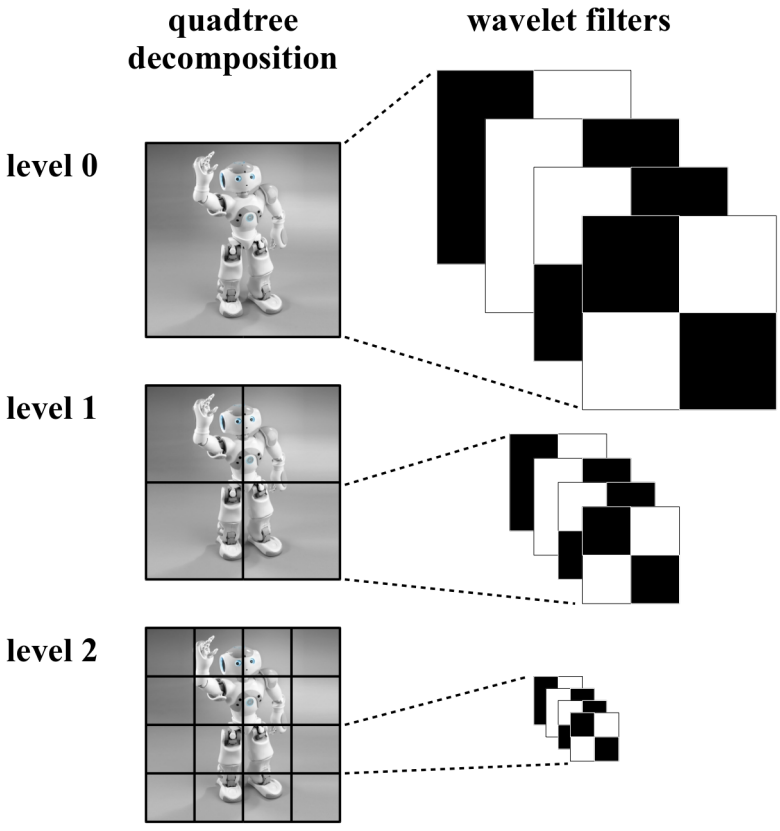


Figure 7.1: Sample quadtree decomposition with Haar wavelet filters. The quadtree is hierarchically decomposed into three levels. Each section is filtered with all of the four shown wavelet filters.

7.1.4 Cluster Joining

The conducted experiments (Chapter 6) showed that M-SOINN was not always able to unite all perceptions of a particular category in one cluster. Especially when clustering the images of a person's face, M-SOINN created separate clusters for the various head orientations that were present in the images. Even for a single person these images differed too much in their original image space and the chosen distance thresholds for joining could not cover this amount of variability. Increasing corresponding threshold values would lead to another problem of merging the inputs of different people into one cluster.

Nevertheless, ICALA did learn relationship information about these clusters by associating the visual inputs with their corresponding labels. This information is encoded in TOSAM where strong associations exist from the images to their corresponding labels and vice versa. All visual inputs that are strongly associated with a specific label are supposed to originate from a single person's face. This association structure makes it obvious as to which perceptions should be actually grouped together. In general, distributed categories can be discovered by checking for strong associations with a single cluster of another modality. But simply creating edges between the corresponding clusters in M-SOINN could result in a large cluster that is likely to attract the inputs of other categories and, thus, lead to over-generalisation. In this regard, it is not always possible to capture the desired similarities with the Euclidean distance in the given image space. Figure 7.2 provides an example illustration of such a situation.

For facilitating a better cluster formation, ICALA needs a projection which maps from the original input space into a feature space. In this feature space, all inputs of one category should be close together (in a Euclidean sense), whereas each pair of inputs that originated from different people should be placed far apart. Hence, the projection should minimise the distance between intra-category instances and maximise the distance between inter-category instances. Alternatively, the projection could minimise the variance for each set of intra-category instances and maximise the variance for each set of inter-category instances.

A suitable projection could be computed using PCA (Bishop, 2007). For the given problem, however, it is important to calculate the covariance matrix on a specific subset of the data. For projecting all the instances of one category as close to each other as possible, the covariance matrix needs to be computed with only these instances. For these instances, the variance in the feature space should be small. The eigenvectors that have the lowest eigenvalues point into the directions with the lowest variance. Thus, some of these dimensions should be considered for spanning the feature space. Similarly, this process must be repeated for each category, each of them extending the feature space by more dimensions. Additionally, samples of instances from all categories should be considered, i.e. only one instance of each

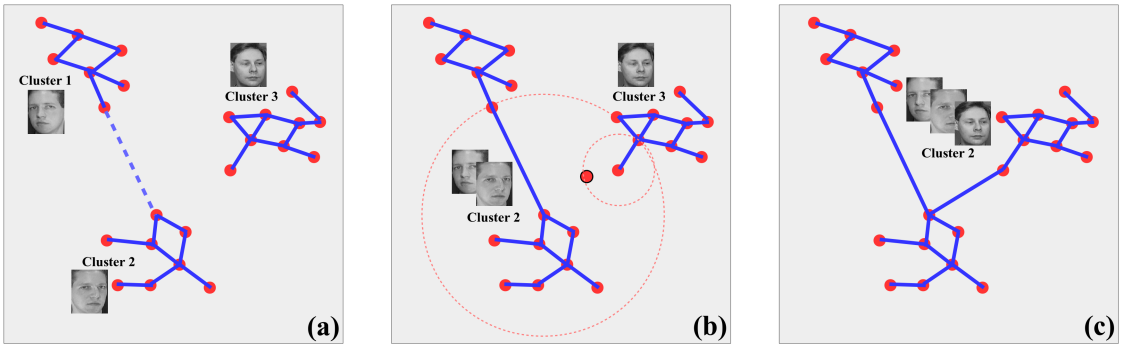


Figure 7.2: Illustration of over-generalisation in M-SOINN when joining clusters based on the associations created in TOSAM. For visualization purposes this illustration considers a topology with only two dimensions. Nodes are shown in red, edges are shown in blue. The images of faces were added as hints to better distinguish between different categories but do not reflect the actual two-dimensional instances. (a) The M-SOINN topology contains three clusters. Clusters 1 and 2 contain instances of one category (e.g. category A), cluster 3 contains instances of another category (e.g. category B). In TOSAM, each cluster is strongly associated with the corresponding category labels (not shown), e.g. "A" and "B". By joining clusters based on the associations in TOSAM, the dotted edge would be created between clusters 1 and 2 because both clusters are associated with label "A". (b) Now all the instances of a single category belong to cluster 2. The recent edge creation resulted in highly increased similarity thresholds of the connected nodes. A new input is received between cluster 2 and cluster 3 (shown in red with a black outline). The two nearest nodes to the input belong to clusters 2 and 3 (corresponding similarity thresholds are shown with dotted circles). As both similarity thresholds are larger than the corresponding distances to the new input, M-SOINN connects the two existing nodes with an edge, instead of creating a new node at the position of the input. (c) The creation of this edge results in a topology with only one single cluster for all instances. This cluster incorrectly contains the instances of different categories.

category. For these instances, the projection should maximise the variance. Hence, the eigenvectors with the largest eigenvalues should be selected and added to the collection of dimensions of the feature space. Finally, such a process would have to be carried out for each modality.

However, further evaluations would be required to decide whether a simple collection of dimensions is adequate for spanning the feature space, or if a more sophisticated method is required to combine these vectors. In particular, the included vectors cannot be assumed to be orthogonal to each other. Again, an evaluation would have to show whether this would have a negative impact in practise or not. Moreover, the approach as described is an offline method and does not work incrementally. When integrated into ICALA, the PCA computation would require a temporary suspension of other processing activities, including the reading of sensory inputs, unless it is computed in the background on a snapshot of the data. As the topology keeps changing over time, the projection needs to be recomputed from time to time, in order to keep it accurate with the most recent data.

Once an appropriate projection has been computed, it can be used to project new inputs into a feature space where another M-SOINN incrementally clusters the incoming data. In this feature space, inputs of one category can be expected to end up in one cluster, as originally desired. To keep the already learned data, M-SOINN needs to recreate clusters for the existing categories. All stored instances, or at least sample instances of each existing cluster, should be given as input to the M-SOINN algorithm that operates on the feature space. Moreover, in order to maintain consistency with TOSAM, corresponding associations would have to be redirected (and merged where necessary) to the clusters in the feature space.

Whereas the given suggestions apply to ICALA as a whole, the following section analyses various aspects of the M-SOINN algorithm alone.

7.2 M-SOINN

7.2.1 Processing Bottlenecks

In terms of the real-time applicability of M-SOINN, the incrementally growing topology can easily accommodate many hundreds of nodes and can become a bottleneck for processing, depending on the dimensionality of the data. An issue can be the cluster joining modification whose computational complexity grows quadratically both with the number of nodes and the number of clusters. Also the refined connection criterion for new nodes has a computational complexity that grows quadratically with the number of nodes in the topology. A general solution here might be to compute the required operations in the background, i.e. to create the new node and proceed with subsequent processing steps while still evaluating the creation of an edge. But as the topology changes in the meantime, the delayed creation of an edge, just as a deferred joining of clusters, can lead to inconsistencies. Any potential problems, such as intermittently deleted nodes, must be taken into account when the described operations should be applied.

But even if these computational bottlenecks can be eliminated, the complexity of M-SOINN still depends linearly on the number of nodes and the dimensionality of the data. Already for determining the nearest node for a new input, the algorithm needs to compute the distances to all existing nodes. These issues could be alleviated with methods for dimensionality reduction (Section 7.1.3). Also by using a divide-and-conquer strategy, the input space could be hierarchically partitioned into smaller areas such that only a subset of nodes would have to be checked for their distance to a new input.

7.2.2 Cluster Evaluation Metrics

For evaluating the clustering performance of M-SOINN, a tailored evaluation measure was used (Section 4.1). The applied score metric considers the desired counts for both categories and clusters and penalises cases where these desired counts have not been reached. The metric also favours smaller variances in these counts over multiple runs. Thus, this scoring measure can only be applied when corresponding extra information about the dataset is available and when the clustering procedure is repeated multiple times. If the number of categories is unknown, other cluster evaluation metrics must be used. Most of these metrics compute a score based on a single clustering result and can be applied without running the algorithm multiple times.

For instance, the *Davies–Bouldin index* (Davies and Bouldin, 1979; Günter and Bunke, 2003) works purely on the obtained clustering result. It takes into consideration the distances from single instances to their respective cluster centroids, as well as the distances between the clusters themselves (Equation 7.1).

$$DBi = \frac{1}{|C|} \sum_{i=1}^{|C|} \max_{i \neq j} \left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right) \tag{7.1}$$

where $|C|$ is the number of clusters, c_i is the centroid of cluster i, σ_i is the average distance from all instances in cluster i to its centroid c_i, and $d(c_i, c_j)$ is the distance between the centroids c_i and c_j. The *Davies–Bouldin index* favours clustering results where the cluster centroids are placed far apart but each cluster for itself is very compact. This scoring preference is reasonable for many clustering scenarios but may be inappropriate when the optimal solution contains non-hyperspherical clusters.

Another evaluation measure is the *Dunn index* (Günter and Bunke, 2003; Bolshakova and Azuaje, 2003). It measures the clustering result based on its weakest point, namely the minimum distance between any two clusters and the maximum intra-cluster distance (Equation 7.2).

$$Di = \min_{1 \leq i \leq |C|} \left\{ \min_{1 \leq j \leq |C|, i \neq j} \left\{ \frac{d(i,j)}{\max_{1 \leq k \leq |C|} d'(k)} \right\} \right\} \tag{7.2}$$

where $|C|$ is the number clusters, $d(i,j)$ is the distance between clusters i and j, and $d'(k)$ is the intra-cluster distance of cluster k. The distance between clusters can be measured by any suitable inter-cluster metric and, similarly, any appropriate intra-cluster metric is adequate for determining the compactness of a cluster. As the *Dunn index* considers only the extreme cases of cluster sparseness and cluster proximity, this measure is prone to outliers in the resulting clustering (Günter and Bunke, 2003).

The described measures may be used for more general evaluations of the M-

SOINN algorithm, or to compare M-SOINN with other clustering algorithms. But as both these measures do not adequately account for non-hyperspherical clusters, and due to the extra information available about the used data (Chapter 4), the introduced evaluation measure was better qualified for ranking the clustering results.

The next section discusses features that are specific to TOSAM, and further shows similarities to empirical findings of psychology research.

7.3 TOSAM

7.3.1 Similarity of Inputs

TOSAM does not include a similarity measure for the patterns of different units. Irrespectively of how much two patterns differ, TOSAM considers all units equally distinct. As a result, TOSAM cannot partially activate other units with similar patterns as the one of the actually activated unit. Also, if a received pattern only slightly differs from a stored pattern, TOSAM is not able to assign it to the latter. ICALA solves this problem by integrating M-SOINN, which groups similar inputs into clusters and only transfers a unique cluster identifier to TOSAM. In case the M-SOINN topology contains multiple clusters for related inputs, then also TOSAM has created corresponding units. But as related inputs can be expected to be perceived in close succession to each other, it is likely that TOSAM has learned associations between these units. So once one of these units is activated, it will spread its activation to associated units and, thus, partially activate them. Under such circumstances, the whole spreading mechanism can be seen as a surrogate for a partial matching method.

7.3.2 Sequence Structures

Specific sequence structures are problematic for TOSAM as, in these cases, the learned stimulus items cannot be recalled properly. For instance, if one stimulus item is repeated within a sequence but followed by different items in each case, TOSAM creates equally strong associations from one single item to two or more other items. This can lead to an overlapping or concurrent recall of these items. This issue also exists when specific stimulus items are used in multiple sequences. In its current state, TOSAM is restricted to sequences without repeated or common stimulus items.

Consider for example the sequence "A" → "X" → "C" → "D" → "X" → "F". When fully and clearly learning this sequence, associations are formed from each item to the corresponding subsequent item. However, from item "X", both "C" and "F" are associated. This creates a problem when spreading activation as in this case "X" spreads equal amounts to both "C" and "F" – leading to a rather weak recall of both these items. The two different situations in which "X" needs to spread

activation can only be distinguished when looking at the context. Here, context means which items were activated before. "A" → "X" should be followed by "C" and "D" → "X" should be followed by "F". If "A", when activated, already spreads part of its activation to "C", and "D" similarly spreads to "F", the resulting activation levels would be different in each situation. In this way, the two situations could be distinguished. Unfortunately, TOSAM in its current configuration does not achieve this effect. Figure 7.3 shows a heatmap plot of the activation levels when recalling the sequence "A" → "B" → "X" → "D" → "E" → "F" → "G" → "X" → "I" → "J".

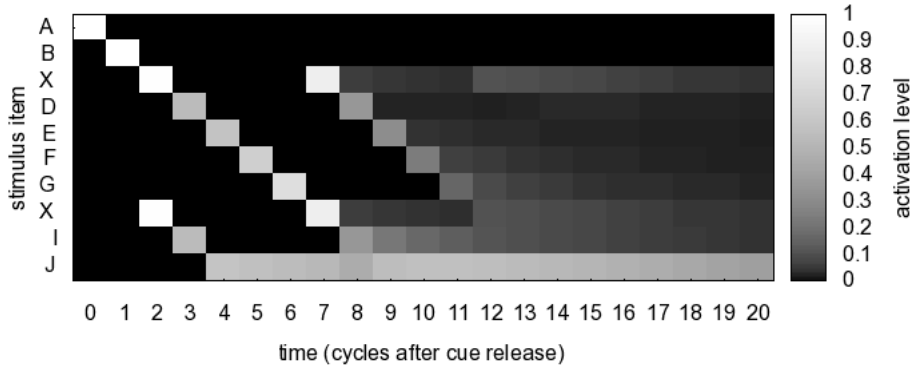


Figure 7.3: The activation levels in all units over several cycles after cue removal when the network learned the 10-stimulus sequence "A" → "B" → "X" → "D" → "E" → "F" → "G" → "X" → "I" → "J" 25 times with a gap of 2 cycles after each stimulus presentation and the cue "A" was shown for 1 cycle. The stimulus item "X" is repeatedly plotted in two rows.

The same problem occurs when two different sequences are to be learned but both have a specific item in common. Once the spreading process reaches this common item, the transmitted signal is split up between the respective subsequent items of both sequences. The recall process continues in parallel on both sequences. Again, the context should have a major influence on the activation state of subsequent units such that both sequences can be recalled separately.

Moreover, if a sequence contains a specific item multiple times in a row, TOSAM does not recall this item multiple times but instead directly proceeds with the next differing item. Allowing self-connectivity in the network could solve this problem but would require changes in the learning rule. Otherwise each unit would increase the connection weight to itself whenever the corresponding stimulus item is shown. These weights would become stronger than the weights to other units, even if the presented sequence did not contain any item in a row.

7.3.3 Processing Bottlenecks

The network of TOSAM can grow dynamically. A new unit is created whenever a novel distinct pattern is received as an input. As the length and the content of a single pattern are not restricted, the network is able to potentially store an infinite number of pattern and thus, grow infinitely large. Of course, the size of the network has implications on the processing time which depends linearly on the number of units and associations. From a certain number of units onwards, TOSAM will no longer be able to maintain the set cycle times. The pruning of units helps to reduce the network size again. But in its current implementation the thresholds for pruning a unit are set to fixed values, which does neither guarantee a maximum number of units nor a maximum limit for the processing time. A better solution here would be to constantly monitor the processing time and adjust the thresholds for pruning, such that more units get pruned when the set cycle time is exceeded but otherwise the thresholds are relaxed to avoid unnecessary pruning.

7.3.4 Spike-Timing-Dependent Plasticity

Spike-Timing-Dependent Plasticity (STDP) describes the effects Long-Term Potentiation (LTP) and Long-Term Depression (LTD) of synapses in relation to the exact timings of pre- and post-synaptic activity (Dan and Poo, 2004; Caporale and Dan, 2008) and, thus, provides an adequate basis for a Hebbian Learning rule with temporally asymmetric learning characteristics.

Several neuroscientific empirical findings (Caporale and Dan, 2008) suggest that a pre-synaptic spike followed by a post-synaptic spike leads to LTP, whereas LTD occurs when a post-synaptic spike happens before a pre-synaptic spike. These effects have been observed in various brain regions including the hippocampus and the visual cortex. Furthermore, experimental findings showed that the potentiation effect is weaker for stronger synapses while the strength of the depression effect remains the same (van Rossum et al., 2000). However, as observed in the hippocampus, at a higher spike firing rate the LTP effect was stronger in magnitude than LTD (Dan and Poo, 2004; Caporale and Dan, 2008). When considering entire bursts of spikes, the inter-burst intervals appear to be more important for synaptic plasticity than the timings of individual spikes (Caporale and Dan, 2008). LTP and LTD are caused by the temporal order of the bursts rather than the exact order of the single spikes (Dan and Poo, 2004).

Such burst-dependent learning effects can be captured by a rate-coded model, which abstracts from the individual spikes but only models the level of spiking activity. TOSAM considers two continuous values for each unit, the activation level and the input load. The input load has an influence on learning and, thus, can be seen as coding the rate of neuronal spiking activity that influences the learning

process. Although the learning rule in TOSAM was not designed with STDP in mind, it resembles several of the empirically observed characteristics. For instance, the magnitude of the induced synaptic change seems to decay exponentially while increasing the gap between two synaptic inputs (van Rossum et al., 2000; Dan and Poo, 2004). Similarly, in TOSAM the exponentially decaying input load influences the weight change such that longer gaps result in smaller changes. According to STDP curves, the occurrence of a post-synaptic stimulus before a pre-synaptic stimulus induces LTD. In TOSAM such stimuli timings lead to a decrease in weight of the involved connection (Chapter 5). Moreover, the amount of weight change in TOSAM is weaker for stronger weights, just as LTP is weaker for stronger synapses (van Rossum et al., 2000). A difference between TOSAM and empirically observed LTP/LTD effects, however, is the exact period of sensitivity to neuronal activity. In the conducted experiments (Chapter 6) the cycle time of TOSAM was set to at least 100 ms and learning could take place with an inter-stimulus interval of a few cycles. But according to empirical findings, the sensitivity for synaptic modification already diminishes at around 50 ms (Song et al., 2000; van Rossum et al., 2000), with a maximum learning strength at around 10 ms (Dan and Poo, 2004). By adjusting some parameters in TOSAM, the model could provide a better fit to observed STDP curves (e.g. with a cycle time of 10 ms and an exponent between 0.6 and 0.7 for the decay of the input load).

Nevertheless, some fundamental improvements could be made in TOSAM, if the aim was to accurately account for the mentioned STDP effect as well as for related findings (Caporale and Dan, 2008). Instead of allowing the association weights to become negative, a biologically more plausible solution would be to let units be either excitatory or inhibitory. Just as there exist excitatory neurons and inhibitory neurons, a unit could have either a positive or a negative influence on the spreading process. Just as a synaptic strength cannot be negative, associations could have only positive weight values. Under these circumstances an inhibitory unit would receive a positive input load and spread its activation over a positively-weighted association but influence the connected unit in an inhibitory way. For excitatory units the spreading process would remain the same. Moreover, the learning process for a specific association could be influenced by the transmitted signal and the input load of the unit which the connection links to. This would also better account for the finding that potentiation and depression of a synaptic weight are influenced by the synaptic potential together with post-synaptic spiking activity (van Rossum et al., 2000).

7.3.5 Primacy and Recency Effect

One major characteristic of TOSAM is that the used learning rule is sensitive to the exact timing of inputs, which allows TOSAM to learn sequential material such as an

ordered list of items. The memorisation and retrieval of list items (usually lexical items such as words) have been studied with human subjects and corresponding recall results were visualized as serial position curves (Anderson, 1999; Crowder and Greene, 2000). These curves show a probability of recall for each list item depending on the item's position within the list. Related results showed two effects that occur with serial order recall, namely the recency effect and the primacy effect. The recency effect refers to the observation that the probability of recalling an item is higher for items towards the end of the list. The primacy effect expresses that items from the beginning of the list are more likely to be recalled.

TOSAM does not aim to accurately reproduce these effects as the model's focus lies more on its general applicability within ICALA. Nevertheless, it is possible to compare the recall results of TOSAM for sequential material (Section 5.2) with the findings reported for serial order recall (Anderson, 1999; Crowder and Greene, 2000). In this case, the activation level of a unit in TOSAM must be interpreted as the probability of recalling the corresponding information. By doing so, the sequential recall process in TOSAM exhibits a primacy effect but no recency effect. The primacy gradient is mainly the result of the activation decay in each unit but is also caused by the signal processing during spreading. The primacy effect does not occur when the sequentially forward-facing associations have near maximum strength. In this case, the signal processing outweighs the activation decay such that the recall probability for each item is very high for the entire list. A recency effect could possibly be achieved by decreasing the activation decay in all units such that later items maintain a high level of activation for a longer time.

7.4 Summary

This chapter discussed several features of M-SOINN, TOSAM as well as ICALA as a whole. In particular, suggestions were made on how the created associations can be used to obtain better clustering results and how potential processing bottlenecks in M-SOINN and TOSAM can be overcome. The chapter also emphasised how the architecture relates to psychological and neuroscientific findings.

Chapter 8

Conclusions

This chapter starts by emphasising the main research contributions of this thesis. Then the achievements of ICALA are presented and the posed research questions are answered, based on the performed evaluations and conducted experiments. Finally, a brief outlook on possible future work is given.

8.1 Contribution Revisited

A learning and memory architecture for robot learning, ICALA, has been developed, which provides a contribution to two research areas: machine learning and robotics. The proposed architecture extends and improves existing machine learning methods and presents a novel integration of associative learning techniques for the use in robots.

8.1.1 Machine Learning

ICALA combines two machine learning methods, M-SOINN and TOSAM, that perform different types of unsupervised learning. When both M-SOINN and TOSAM are used in conjunction within ICALA, they group and structure the inputs both spatially and temporally. The architecture can incrementally learn from incoming information by directly using the captured sensory data while being tolerant to noise in the input patterns and allowing a flexible input timing (Chapter 3). This functionality can be realised due to key features of the included machine learning methods and their combined usage within ICALA. The importance of corresponding features is emphasized in the following.

- ICALA performs incremental clustering and tolerates noise in the inputs.

  In contrary to clustering algorithms like *k-Means* or *Expectation-Maximization* (Bishop, 2007; Witten et al., 2011), with M-SOINN the number of clusters does not need to be decided in advance. The topology is learned incrementally,

which eliminates the requirement of having to know the whole dataset in advance. Furthermore, M-SOINN does not require each input to be within a specific numeric range. This is beneficial for incremental learning tasks where normalisation of the input data can be a problem as the entire training dataset is not known in advance. Moreover, the algorithm can represent clusters of any arbitrary shape and is, in particular, not restricted to hyperspherical clusters. With the formation of these clusters the recognition process becomes tolerant to noise in the input patterns (Section 6.1).

The M-SOINN algorithm works with only one network layer and does not require a separate learning phase as SOINN for learning a second layer. This makes M-SOINN suitable for online learning tasks. M-SOINN supports additional refinement operations that proved beneficial in the evaluation of the algorithm (Chapter 4). Although no single best modification combination could be identified, the inclusion of modifications could always help the algorithm produce better clustering solutions than without the modifications. Thus, they constitute valuable additions to this topology-based clustering approach.

- ICALA uses a dynamically growing storage structure and allows a flexible timing of the perceived inputs.

  TOSAM is a recurrent neural network model in which activation flows between units within a single fully-connected layer. Instead of using a network of a fixed size, the TOSAM network can grow dynamically. This eliminates any theoretical storage limit as could be imposed by a fixed-size network. TOSAM learns incrementally such that new information can be appended while the learned information is already available for recall (Section 6.2).

  TOSAM extends the standard Hebbian learning rule in order to account for the exact timing of the involved stimuli. Combined with a network connectivity with two directed associations between each pair of units, TOSAM can form asymmetric associations between stimuli. In this way, TOSAM can learn about the sequential order of multiple inputs in a row without requiring a strict protocol for the presentation of the inputs (Chapter 5). One-to-one and one-to-many associations can be learned without any further modifications (Section 6.1). Moreover, the learning rule tolerates short gaps between the perceived stimuli. For any key-response pair of inputs, it is not required that the key is directly followed by the response (i.e. in the next processing cycle). This enables the asynchronous operation of TOSAM and M-SOINN modules and, in particular, the use of different cycle times (Chapter 6).

8.1.2 Robotics

Programming a robot to execute different behaviours can be a tedious task (Saunders et al., 2006), especially when relying on manual programming to extend the robot's behaviours repertoire. Even though reactive behaviours can be created on the basis of a set of declarable rules and actions, each of them must be manually defined in this case. Moreover, expert knowledge is required to implement each additional functionality. Under these circumstances a non-specialist human is strictly limited to using the robot with only the already available behaviours.

- ICALA avoids the requirement of technical knowledge and can learn from the received sensory inputs alone.

  A robot companion should be able to learn from the interaction with humans (Dautenhahn, 2004; Castellano et al., 2008). This requires not only methods for learning and adaptation, but also some means of extracting the required information from the interaction itself – in contrary to directly engineering more data into the architecture of the robot. The experiment in Section 6.1 showed how ICALA learns new associations by using the data captured by the robots sensors. Without having to know technical details of the architecture, a user can easily extend the robot's set of behaviours by presenting new inputs, such as a new visual pattern together with a desired arm pose.

  At the same time, ICALA offers enough flexibility for a more technically-skilled person to customise the basic set of input and output modalities. The modular structure of the architecture allows to configure multiple modules based on the availability of sensors and actuators. Also artificial inputs from other data sources can be added. A favourable side effect of the architecture's modular design is that ICALA is not restricted to be used on one single robot platform. Instead, the architecture could be the central driving component for various robot platforms with a multitude of different configurations. As long as sensory data can be made available to an M-SOINN module, it can be integrated into ICALA.

8.2 Achievements

ICALA is an architecture for associative learning with applicability to robots. The focus of ICALA is on the incremental and unsupervised nature of the used learning methods. Both M-SOINN and TOSAM start with empty knowledge bases in form of empty networks that grow while new inputs are received by the architecture. Moreover, ICALA neither employs any methods that are specific to a certain type of inputs like images of faces, nor does it use any other hard-coded domain information.

The capacity of accumulating more and more information over time was confirmed in the experiment on learning the faces of 40 different people (Section 6.2). The topology of nodes in M-SOINN as well as the network of units in TOSAM were growing incrementally. In every moment the acquired information was ready to be recalled, which was tested in regular intervals. Although TOSAM could not always recall the correct text label, M-SOINN was able to store clusters for nearly all included people.

Neither M-SOINN nor TOSAM require the declaration of goals or class attributes as used in classification tasks. M-SOINN performs the task of clustering while TOSAM uses the relative timing information about the incoming inputs to learn corresponding relationships. Thus, both methods work fully unsupervised. When used in a robot, ICALA can learn about the robot's environment without further external intervention. This ability was illustrated in the experiment described in Section 6.2, where the robot perceived the images of people's faces together with their corresponding text labels. The latter are artificial inputs that could not have been obtained directly by the robot's sensors. However, instead of text labels, these inputs could have consisted of certain characteristics of the corresponding person's voice, such as pitch and tone – if such data existed for the given images. In a setup with actual people standing in front of the robot, the voice could be picked up by the robot's microphone. In any case, ICALA can autonomously create internal mappings between the perceptions of every single person and, in this way, structure the incoming data.

While this form of learning does not require a controlled environment, ICALA can also learn only desired mappings, i.e. when the presented inputs are carefully selected. The experiment on the association of road signs and arm poses (Section 6.1) demonstrated this possibility by teaching a robot various stimulus-response behaviours. A visual stimulus was presented to be assigned to a kinematic response. For learning situations like these an external teacher is required. Thus, the learning in ICALA can also happen in a supervised form while the algorithms still operate completely unsupervised.

Sensory data is the only source of information for ICALA. In both experiments, the architecture used the sensory data from the robot as provided by the NAOqi API[1]. This was the array of pixel values for any visual inputs and the reading of joint angles from the robot's arms. The received values do not undergo any specialised preprocessing steps like edge detection, the extraction of facial features or the detection of predefined arm poses. The processing steps in M-SOINN are generic and work with any multi-dimensional vector data while the distance information between nodes is sufficient for most processing steps. TOSAM can learn and store the relationships between any discretized input patterns of an arbitrary length

[1] `http://doc.aldebaran.com/1-14/naoqi/`

without making use of an ontology or any other human-crafted information source. Fundamental for the learning process in TOSAM is the co-occurrence and timing of inputs, which is implicitly provided when perceiving an input.

In conclusion, ICALA fulfils several important demands of robot learning and allows to answer the formulated research questions.

8.2.1 Answers to Research Questions

1. How can a versatile learning and memory architecture be constructed without requiring domain-specific processing or relying on domain-specific information?

   This thesis proposed ICALA, a versatile learning and memory architecture that avoids any domain-specific processing. ICALA does not need any domain-specific information about the inputs nor additional expert knowledge. All inputs are treated as real-valued patterns of data and are processed in the same way. The versatility of the architecture was demonstrated in the conducted experiments (Chapter 6), where ICALA successfully handled visual data containing road signs, visual data containing faces, motor-control data containing arm joint angles as well as enumerated textual label inputs.

2. How can such an architecture be applied to robot learning while the robot's sensors are the only source of information?

   ICALA obtains all information solely via the available sensors of the robot. The received inputs are incrementally grouped into clusters of similar inputs. The formation of clusters makes the recognition process robust to noise which can be inherent in the sensory data, e.g. in the visual patterns captured by a camera (Section 6.1). It is not required to manually label these clusters. ICALA learns relationship information based entirely on the co-occurrence of the inputs. The architecture does not rely on human-crafted knowledge bases such as ontologies. The algorithms used within ICALA work fully unsupervised.

3. Which type of behaviours can a robot learn while solely relying on generic unsupervised learning methods?

   This thesis proved that by using only generic unsupervised learning algorithms, a robot is able to learn stimulus-response behaviours, or more generally, reactive behaviours. ICALA does not have the capability to make complex plans, neither does ICALA set internal goals and tries to fulfil them. The entire learning process in ICALA is not oriented towards specific goals and, thus, undirected. However, learning can be both directed and unsupervised at the same time if the goals are chosen based on abstract concepts such as novelty or learning progress. Also the simulation of emotional states could influence the immediate response of the robot and lead to more complex behavioural patterns. Integrating such

approaches into ICALA remains future work which will be discussed in the following section.

In consideration of the outcome regarding each of the research questions, the stated hypothesis is accepted: *A learning and memory architecture that uses only unsupervised learning methods without any domain knowledge allows a robot to learn reactive behaviours.*

8.3 Future Work

The research described here offers various opportunities for future research. While several new functionalities could be added, it is also necessary to tackle the issues present in the current implementation of ICALA as discussed in Chapter 7. In particular the integration of methods for dimensionality reduction could mitigate the decreasing runtime performance under heavy load. Also desirable would be experiments with an architecture configuration with more modalities. By including additional input/output modalities, potentially more complex behavioural patterns could emerge due to the richness and variability of internal representations. Additionally, it would be possible to generate more elaborated behaviours by enhancing the mechanisms for learning and recall, which will be described in the following sections.

8.3.1 Learning Enhancements

In its current state TOSAM uses the same learning rate for every association in its network. A reasonable extension would be to allow individual and dynamic learning rates such that the sensitivity of each association could change independently. Existing models for classical conditioning (Le Pelley, 2004) considered this aspect in a similar way by assigning an associability value to each cue stimulus. If implemented in TOSAM, the model could potentially account for the learning phenomena sensitization and habituation (Anderson, 1999). If a cue is a poor predictor for another immediately following stimulus, this pair of stimuli can be assumed to be rather novel. To quickly memorise this new association, the corresponding associability should be high initially. Once this stimulus pair has been frequently observed and the association has been learned, a lower associability should reflect the habituation towards the respective stimulus. Over time the associability would increase again and the habituation effect would diminish. In this way, the unlearning process of TOSAM would have a weaker effect on associations for regularly observed stimulus pairs while allowing faster unlearning for rarely observed pairs. When perceiving an unexpected following stimulus, associations with a low associability would be less prone to unlearning. However, for verifying these assumptions, the actual influence of such an addition would have to be thoroughly evaluated.

As the learning process in ICALA shares similarities with classical conditioning, it seems possible to include aspects of instrumental conditioning as well. In instrumental conditioning an organism learns that in a given context (stimulus situation) a particular action (response) will lead to a reinforcement (Anderson, 1999). Depending on whether this reinforcement is desirable for the organism or not, the corresponding action is more likely to be performed or avoided. Thus, by selecting an appropriate reinforcement, certain actions can be either provoked or suppressed. With such a mechanism ICALA could learn from feedback and, in this way, the learning process could be directed towards specific behaviours. However, the effects of possible reinforcers need to be clearly defined, i.e. which reinforcers are desirable and which are not? Also taken into consideration should be the methods for reinforcement learning described by Sutton and Barto (1998). Although it is not possible to directly apply the described algorithms to ICALA, a transfer of corresponding principles seems reasonable and would enhance the architecture's learning capabilities.

8.3.2 Explorative Behaviours

A possible way of directing the learning process without requiring supervised learning methods is by setting abstract goals for the robot. Such an abstract goal could be to maximise the learning progress. Alternatively, the robot could be driven by curiosity in order to stimulate explorative behaviours.

Blank et al. (2005) suggested to include the principle of self-motivation in an architecture for developmental learning. A corresponding mechanism would lead the architecture to explore novel situations and, by learning about more and more situations, increase the architecture's competency and overall performance. When used in a robot, the robot would get bored by easily predictable situations and start exploring its surroundings to discover new situations. At the same time, the robot would avoid environments that are too chaotic and too hard to correctly make predictions about. According to this idea, a robot is driven by the novelty of inputs and shows, in this sense, artificial curiosity. Moreover, robot behaviours are created without having to set a discrete learning goal.

Oudeyer et al. (2007) proposed an approach for autonomously learning robots, which is based on reinforcement learning. Their system learns about the relationship between perceived world states as a consequence of specific robot actions, and tries to predict future states based on the acquired knowledge. Most importantly, the system does not simply try to minimise its prediction error. In static environments an error-free prediction can be easily achieved by remaining idle, such that the perceived world state does not change at all. This state could be predicted perfectly while no other situations are explored. But instead, the action selection process is driven by the more abstract goal of maximising the change in prediction error. Thus, the robot is always eager to learn more. Once a situation is accurately predictable, no learning

progress can be made. Consequently, the robot will execute actions to change the situation to a more unpredictable one. If, on the other hand, a particular situation is highly unpredictable and the error remains high, no learning progress can be made either and the robot will search for situations that are easier to learn. In this way, the learning challenges are chosen by the robot itself and not by an external tutor. Overall, the approach of Oudeyer et al. (2007) allows robots to learn actively and exploratively while still being unsupervised.

8.3.3 Attention Mechanism

An attention mechanism for ICALA would be desirable as it could help to reduce the amount of data that is processed by the entire architecture. Especially if an architecture configuration includes many modules that receive inputs from many modalities, not all the data may be important at any given time. Depending on the context and situation, the inputs from different sources (e.g. different sensors) could be weighted differently. This would allow ICALA to focus and direct its attention by automatically reallocating its available perceptual resources. Also specific patterns may have a higher or lower importance, depending on the current context. An attention mechanism could help to adequately select the most relevant inputs.

Ideas for a general attention mechanism are given by Helgason et al. (2014), who demand such a mechanism to be data-driven, employ fine-grained structures, have predictive capabilities, use a unified sensory pipeline and work with symbolic knowledge representations. The approach suggested by Helgason et al. (2014) uses data items and processes as its main data units and three components to influence the priorities for incoming data. A goal-driven data prioritizer positively biases data patterns that are required to fulfil current goals, a novelty-driven data prioritizer prefers unseen and unexpected information, and an experience-driven process prioritizer considers previous contributions of data patterns towards goals and sets more attention towards patterns with a high utility (Helgason et al., 2014). Similar components could be implemented in ICALA and would, in particular, influence the activation spreading in TOSAM. However, the important question of where the goals originate from remains unanswered. For including a goal-driven component, a separate mechanism for establishing goals would have to be investigated.

8.3.4 Emotional State Module

Research on human memory has shown that both the acquisition of knowledge and the retrieval are dependent on circumstances like the physiological state of the body and the current emotional state (Baddeley et al., 2009). For instance, the rate of retrieval for certain memories is higher when being in the mood in which the corresponding information has been perceived and encoded (mood-dependent

memory). Furthermore, human memory is more likely to retrieve memories associated with a certain mood when actually being in this mood (mood-congruent memory). If the person is sad, he or she is more likely to retrieve sad memories than happy ones (Baddeley et al., 2009). Emotions can be seen as a separate input being fed into the architecture. Due to the association of units in TOSAM, other inputs would be linked to specific emotional states. At the same time, the corresponding units could be used to compute an output which would influence the actual emotional state.

Instead of using discrete emotional states, various hormone levels could be simulated. Compositions of all these levels together would represent different emotional states. The hormone levels could be manipulated by simulated pain or pleasure. For instance, inputs from a robot's bumper sensors could represent pain whereas the pressure sensors on a (humanoid) robot's head could detect pleasant stroking movements. However, these relationships would have to be hard-coded and, thus, be considered as simply given as a consequence of the embodiment of the robot. Corresponding inputs could provide the architecture with feedback about whether certain actions have a positive or negative effect on the robot's emotional or physical state. This would present a step towards goal-oriented behaviour because the robot could deliberately avoid actions with a negative outcome. Such a functionality requires modifications on the spreading mechanism in TOSAM such that the units linked to unhealthy body states permanently inhibit strongly associated units. In turn, explicit information would be required about which states are to be avoided and which are the desired ones.

8.4 Summary

This chapter described how ICALA contributes to the domains of machine learning and robotics, emphasised achievements of the approach and gave directions for future research. Throughout the entire thesis a major focus was put on keeping the learning and recall methods general enough and on avoiding any hard-coded knowledge. Also potential extensions of ICALA should follow this guideline as the ability to autonomously learn from the environment appears to be highly important for truly versatile robot companions.

References

Akgun, Baris; Cakmak, Maya; Jiang, Karl, and Thomaz, Andrea L. Keyframe-based Learning from Demonstration. *International Journal of Social Robotics*, 4(4): 343–355, 2012.

Alahakoon, Damminda; Halgamuge, Saman K., and Srinivasan, Bala. A Self Growing Cluster Development Approach to Data Mining. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, volume 3, pages 2901–2906, 1998.

Alahakoon, Damminda; Halgamuge, Saman K., and Srinivasan, Bala. Dynamic Self-Organizing Maps with Controlled Growth for Knowledge Discovery. *IEEE Transactions on Neural Networks*, 11(3):601–614, 2000.

Amor, Heni Ben; Berger, Erik; Vogt, David, and Jung, Bernhard. Kinesthetic Bootstrapping: Teaching Motor Skills to Humanoid Robots through Physical Interaction. In Mertsching, Bärbel; Hund, Marcus, and Aziz, Zaheer, editors, *KI 2009: Advances in Artificial Intelligence*, volume 5803 of *Lecture Notes in Computer Science*, pages 492–499. Springer Berlin Heidelberg, 2009.

Anderson, John R. A Spreading Activation Theory of Memory. *Journal of Verbal Learning and Verbal Behavior*, 22(3):261—295, 1983.

Anderson, John R. *Learning and Memory: An Integrated Approach*. John Wiley & Sons, 2nd edition, 1999.

Anderson, John R. and Lebiere, Christian. *The Atomic Components of Thought*. Lawrence Erlbaum Associates, 1998.

Anderson, John R.; Bothell, Daniel; Byrne, Michael D.; Douglass, Scott; Lebiere, Christian, and Qin, Yulin. An Integrated Theory of the Mind. *Psychological Review*, 111(4):1036–1060, 2004.

Arkin, Ronald C. *Behavior-based Robotics*. MIT Press, 1998.

Aylett, Ruth; Castellano, Ginevra; Raducanu, Bogdan; Paiva, Ana, and Hanheide, Marc. Long-term Socially Perceptive and Interactive Robot Companions: Chal-

lenges and Future Perspectives. In *Proceedings of the 13th International Conference on Multimodal Interfaces*, pages 323–326, 2011.

Baddeley, Alan; Eysenck, Michael W., and Anderson, Michael C. *Memory*. Psychology Press, 2009.

Baddeley, Alan D. *Working Memory*. Oxford Psychology Series. Oxford University Press, 1986.

Baddeley, Alan D. *Human Memory: Theory and Practice*. Psychology Press, revised edition, 1997.

Balkenius, Christian and Morén, Jan. Computational Models Of Classical Conditioning: A Comparative Study. Technical report, Lund University Cognitive Science, 1998.

Bauer, Hans-Ulrich and Villmann, Thomas. Growing a Hypercubical Output Space in a Self-Organizing Feature Map. *IEEE Transactions on Neural Networks*, 8(2): 218–226, 1997.

Baxter, Paul E.; de Greeff, Joachim, and Belpaeme, Tony. Cognitive architecture for human-robot interaction: Towards behavioural alignment. *Biologically Inspired Cognitive Architectures*, 6:30–39, 2013.

Bear, Mark F.; Connors, Barry W., and Paradiso, Michael A. *Neuroscience: Exploring the Brain*. Lippincott Williams and Wilkins, 3rd edition, 2006.

Bekey, George A. *Autonomous Robots: From Biological Inspiration to Implementation and Control*. MIT Press, 2005.

Bishop, Christopher M. *Pattern Recognition and Machine Learning*. Springer, 2007.

Blank, Douglas; Kumar, Deepak; Meeden, Lisa, and Marshall, James B. Bringing up robot: Fundamental mechanisms for creating a self-motivated, self-organizing architecture. *Cybernetics and Systems*, 36(2):125–150, 2005.

Bolshakova, Nadia and Azuaje, Francisco. Cluster validation techniques for genome expression data. *Signal Processing*, 83:825–833, 2003.

Botvinick, Matthew M. and Plaut, David C. Short-term memory for serial order: A recurrent neural network model. *Psychological Review*, 113(2):201–233, 2006.

Bouchachia, Abdelhamid. Evolving Clustering: An Asset for Evolving Systems. IEEE SMC – eNewsletter, September 2011. Issue #36.

Boyer, Maud; Destrebecqz, Arnaud, and Cleeremans, Axel. Processing abstract sequence structure: learning without knowing, or knowing without learning? *Psychological Research*, 69:383–398, 2005.

Burgess, Neil and Hitch, Graham J. Memory for serial order: A network model of the phonological loop and its timing. *Psychological Review*, 106(3):551–581, 1999.

Burgess, Neil and Hitch, Graham J. A revised model of short-term memory and long-term learning of verbal sequences. *Journal of Memory and Language – Special Issue on Memory Models*, 55(4):627—-652, 2006.

Caporale, Natalia and Dan, Yang. Spike Timing–Dependent Plasticity: A Hebbian Learning Rule. *Annual Review of Neuroscience*, 31:25–46, 2008.

Castellano, Ginevra; Aylett, Ruth; Dautenhahn, Kerstin; Paiva, Ana; McOwan, Peter W., and Ho, Steve. Long-term affect sensitive and socially interactive companions. In *Proceeding of the Fourth International Workshop on Human-Computer Conversation*, 2008.

Cleeremans, Axel and Dienes, Zoltán. Computational Models of Implicit Learning. In Sun, Ron, editor, *The Cambridge Handbook of Computational Psychology*, chapter 14, pages 396–421. Cambridge University Press, 2008.

Cleeremans, Axel; Destrebecqz, Arnaud, and Boyer, Maud. Implicit learning: news from the front. *Trends in Cognitive Sciences*, 2(10):406–416, October 1998.

Crannell, C. W. and Parrish, J. M. A Comparison of Immediate Memory Span for Digits, Letters, and Words. *The Journal of Psychology: Interdisciplinary and Applied*, 44(2):319–327, 1957.

Crowder, Robert G. and Greene, Robert L. Serial Learning: Cognition and Behavior. In Tulving, Endel and Craik, Fergus I. M., editors, *The Oxford Handbook of Memory*, chapter 8, pages 125–135. Oxford University Press, 2000.

Dan, Yang and Poo, Mu-ming. Spike Timing-Dependent Plasticity of Neural Circuits. *Neuron*, 44:23–30, 2004.

Dautenhahn, Kerstin. Robots We Like to Live With?! - A Developmental Perspective on a Personalized, Life-Long Robot Companion. In *Proceedings of the 2004 IEEE International Workshop on Robot and Human Interactive Communication*, pages 17–22, 2004.

Dautenhahn, Kerstin; Woods, Sarah; Kaouri, Christina; Walters, Michael L.; Koay, Kheng Lee, and Werry, Iain. What is a Robot Companion – Friend, Assistant or Butler? In *Proceedings of the IEEE IRS/RSJ International Conference on Intelligent Robots and Systems*, 2005.

Davies, David L. and Bouldin, Donald W. A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):224–227, 1979.

de Castro, Leandro N. and Timmis, Jon I. Artificial immune systems as a novel soft computing approach. *Soft Computing*, 7(8):526–544, 2003.

de Castro, Leandro N. and Timmis, Jonathan. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer, 2002.

De Silva, Daswin; Alahakoon, Damminda, and Dharmage, Shyamali. Cluster Analysis using the GSOM: Patterns in Epidemiology. In *Proceedings of the Third International Conference on Information and Automation for Sustainability*, pages 63–69, 2007.

Dienes, Zoltan. Connectionist and Memory-Array Models of Artificial Grammar Learning. *Cognitive Science*, 16:41–79, 1992.

Dittenbach, Michael; Merkl, Dieter, and Rauber, Andreas. The Growing Hierarchical Self-Organizing Map. In *Proceedings of the International Joint Conference on Neural Networks*, volume 6, pages 15–19, 2000.

Dittenbach, Michael; Rauber, Andreas, and Merkl, Dieter. Uncovering hierarchical structure in data using the growing hierarchical self-organizing map. *Neurocomputing*, 48:199–216, 2002.

Domingos, Pedro. A Few Useful Things to Know about Machine Learning. *Communications of the ACM*, 55(10):78–87, 2012.

Du, K.-L. Clustering: A neural network approach. *Neural Networks*, 23(1):89—-107, 2010.

Ebbinghaus, Hermann. *Memory: A Contribution to Experimental Psychology*. Teachers College, Columbia University, 1913.

Elman, Jeffrey L. Finding Structure in Time. *Cognitive Science*, 14:179–211, 1990.

Eysenck, Michael W. and Keane, Mark. *Cognitive Psychology: A Student's Handbook*. Psychology Press, 5 edition, 2005.

Farrell, Simon and Lewandowsky, Stephan. An endogenous distributed model of ordering in serial recall. *Psychonomic Bulletin & Review*, 9(1):59–79, 2002.

French, Robert M. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, April 1999.

Fritzke, Bernd. A Growing Neural Gas Network Learns Topologies. In Tesauro, Gerald; Touretzky, David, and Leen, Todd, editors, *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press, 1995.

Fritzke, Bernd. A Self-Organizing Network That Can Follow Non-stationary Distributions. In *Proceedings of ICANN'97: International Conference on Artificial Neural Networks*, pages 613–618. Springer, 1997.

Frixione, Marcello and Lieto, Antonio. Exemplars, Prototypes and Conceptual Spaces. In Chella, Antonio; Pirrone, Roberto; Sorbello, Rosario, and Jóhannsdóttir, Kamilla Rún, editors, *Biologically Inspired Cognitive Architectures 2012*, volume 196 of *Advances in Intelligent Systems and Computing*, chapter Advances in Intelligent Systems and Computing, pages 131–136. Springer Berlin Heidelberg, 2013.

Furao, Shen and Hasegawa, Osamu. An incremental network for on-line unsupervised classification and topology learning. *Neural Networks*, 19(1):90–106, 2006.

Furao, Shen; Ogura, Tomotaka, and Hasegawa, Osamu. An enhanced self-organizing incremental neural network for online unsupervised learning. *Neural Networks*, 20: 893—903, 2007.

Gais, S. and Born, J. Declarative memory consolidation: Mechanisms acting during human sleep. *Learning & Memory*, 11(6):679–685, 2004.

Graziano, Vincent; Koutník, Jan, and Schmidhuber, Jürgen. Unsupervised Modeling of Partially Observable Environments. In Gunopulos, Dimitrios; Hofmann, Thomas; Malerba, Donato, and Vazirgiannis, Michalis, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 6911 of *Lecture Notes in Computer Science*, pages 503–515. Springer Berlin Heidelberg, 2011.

Grossberg, Stephen. Competitive Learning: From Interactive Activation to Adaptive Resonance. *Cognitive Science*, 11:23–63, 1987.

Grossberg, Stephen. Adaptive Resonance Theory: How a brain learns to consciously attend, learn, and recognize a changing world. *Neural Networks*, 37:1–47, 2013.

Günter, Simon and Bunke, Horst. Validation indices for graph clustering. *Pattern Recognition Letters*, 24:1107—-1113, 2003.

Haikonen, Pentti O. A. The Role of Associative Processing in Cognitive Computation. *Cognitive Computation*, 1(1):42–49, 2009.

Haikonen, Pentti O. A. XCR-1: An Experimental Cognitive Robot Based on an Associative Neural Architecture. *Cognitive Computation*, 3:360–366, 2011.

Harnad, Stevan. The Symbol Grounding Problem. *Physica D: Nonlinear Phenomena*, 42:335—-346, 1990.

Haykin, Simon. *Neural Networks and Learning Machines*. Prentice Hall, 3rd edition, 2008.

Hebb, Donald O. *The Organization of Behavior – A Neuropsychological Theory*. John Wiley & Sons, 1949.

Helgason, Helgi Pall; Thorisson, Kristinn R., and Garrett, Deon. Towards a General Attention Mechanism for Embedded Intelligent Systems. *International Journal of Computer Science and Artificial Intelligence*, 4(1):1–7, 2014.

Ho, Wan Ching; Dautenhahn, Kerstin, and Nehaniv, Chrystopher L. Computational memory architectures for autobiographic agents interacting in a complex virtual environment: a working model. *Connection Science*, 20(1):21–65, 2008.

Ho, Wan Ching; Lim, Mei Yii; Vargas, Patricia A.; Enz, Sibylle; Dautenhahn, Kerstin, and Aylett, Ruth. An Initial Memory Model for Virtual and Robot Companions Supporting Migration and Long-term Interaction. In *Proceedings of the 18th IEEE International Symposium on Robot and Human Interactive Communication*, pages 277–284, 2009.

Hopfield, John J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79(8):2554–2558, April 1982.

Hopfield, John J. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences of the United States of America*, 81(10):3088–3092, May 1984.

Husbands, Phil; Smith, Tom; Jakobi, Nick, and O'Shea, Michael. Better Living Through Chemistry: Evolving GasNets for Robot Control. *Connection Science*, 10(3–4):185–210, 1998.

Kahana, Michael J. Associative retrieval processes in free recall. *Memory & Cognition*, 24(1):103–109, 1996.

Kahana, Michael J. Associative symmetry and memory theory. *Memory & Cognition*, 30(6):823–840, 2002.

Kahana, Michael J. and Caplan, Jeremy B. Associative asymmetry in probed recall of serial lists. *Memory & Cognition*, 30(6):841–849, 2002.

Ke, Yan; Hoiem, Derek, and Sukthankar, Rahul. Computer Vision for Music Identification. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 597–604, 2005.

Keysermann, Matthias U. ICALA: Incremental Clustering and Associative Learning Architecture. In *Adaptive and Intelligent Systems*, volume 8779 of *Lecture Notes in Computer Science*, pages 70–79, 2014. doi:10.1007/978-3-319-11298-5_8.

Keysermann, Matthias U. and Vargas, Patricia A. Desiderata for a Memory Model. In *Proceedings of the 12th UK Workshop on Computational Intelligence*, pages 37–44, 2012.

Keysermann, Matthias U. and Vargas, Patricia A. An online learning system for autonomously operating robots based on spatial and temporal associative learning. In *International IEEE/EPSRC Workshop on Autonomous Cognitive Robotics*, University of Stirling, Stirling, UK, 2014a.

Keysermann, Matthias U. and Vargas, Patrícia A. A learning and memory architecture for robot companions based on incremental associative learning. In *Proceedings of the IEEE RO-MAN '14 Workshop on Developmental and bio-inspired approaches for memory and emotion modelling in cognitive robotics*, pages 13–14, 2014b.

Keysermann, Matthias U. and Vargas, Patrícia A. Towards Autonomous Robots Via an Incremental Clustering and Associative Learning Architecture. *Cognitive Computation*, 7(4):414–433, 2015. doi:10.1007/s12559-014-9311-y.

Klopf, Harry A. A neuronal model of classical conditioning. *Psychobiology*, 16(2): 85–125, 1988.

Kohonen, Teuvo. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, 3rd edition, 2001.

Koskela, Timo; Varsta, Markus; Heikkonen, Jukka, and Kaski, Kimmo. Temporal Sequence Processing using Recurrent SOM. In *Proceedings of the Second International Conference on Knowledge-Based Intelligent Electronic Systems*, volume 1, pages 290–297, 1998.

Kosko, Bart. Differential Hebbian Learning. *AIP Conference Proceedings*, 151: 277–282, 1986.

Koutnik, Jan. Inductive Modelling of Temporal Sequences by Means of Self-organization. In *Proceeding of the Internation Workshop on Inductive Modelling*, pages 269–277, 2007.

Koutník, Jan and Šnorek, Miroslav. Temporal Hebbian Self-Organizing Map for Sequences. In Kůrková, Véra; Neruda, Roman, and Koutník, Jan, editors, *Artificial Neural Networks – ICANN 2008*, volume 5163 of *Lecture Notes in Computer Science*, pages 632–641. Springer Berlin Heidelberg, 2008.

Laird, John E. *The Soar Cognitive Architecture*. MIT Press, 2012.

Langley, Pat; Laird, John E., and Rogers, Seth. Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, 10(2):141–160, 2009. ISSN 1389-0417.

Le Pelley, M. E. The role of associative history in models of associative learning: A selective review and a hybrid model. *The Quarterly Journal of Experimental Psychology*, 57(3):193–243, 2004.

Lewandowsky, Stephan and Murdock Jr., Bennet B. Memory for serial order. *Psychological Review*, 96(1):25–57, 1989.

Li, Shu-Chen and Lewandowsky, Stephan. Forward and Backward Recall: Different Retrieval Processes. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 21(4):837–847, 1995.

Lim, Mei Yii; Ho, Wan Ching, and Aylett, Ruth. Spreading Activation - An Autobiographic Memory Retrieval Mechanism for Social Companions. In *Proceedings of the 10th International Conference on Intelligent Virtual Agents*, 2010.

Lim, Mei Yii; Aylett, Ruth; Vargas, Patricia A.; Enz, Sibylle, and Ho, Wan Ching. Forgetting Through Generalisation - A Companion with Selective Memory. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, volume 3, pages 1119–1120, 2011.

Mahadevan, Sridhar. Machine Learning for Robots: A Comparison of Different Paradigms. In *Proceedings of the Workshop on Towards Real Autonomy, IEEE/RSJ Internaltional Conference on Intelligent Robots and Systems*, 1996.

Malt, Barbara C. An On-line Investigation of Prototype and Exemplar Strategies in Classification. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 15(4):539–555, 1989.

Marsland, Stephen; Shapiro, Jonathan, and Nehmzow, Ulrich. A self-organising network that grows when required. *Neural Networks*, 15(8):1041–1058, 2002.

Martinetz, Thomas and Schulten, Klaus. Topology Representing Networks. *Neural Networks*, 7(3):507–522, 1994.

Martinetz, Thomas M.; Berkovich, Stanislav G., and Schulten, Klaus J. "Neural-Gas" Network for Vector Quantization and its Application to Time-Series Prediction. *IEEE Transactions on Neural Networks*, 4(4):558–569, 1993.

Minton, Steven. Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, 42(2—3):363–391, 1990.

Murdock Jr., Bennet B. A distributed memory model for serial-order information. *Psychological Review*, 90(4):316–338, 1983.

Murphy, Robin R. *Introduction to AI Robotics*. MIT Press, 2000.

Neal, Mark and Labrosse, Fred. Rotation-invariant appearance based maps for robot navigation using an artificial immune network algorithm. In *Proceedings of the Congress on Evolutionary Computation*, volume 1, pages 863–870, 2004.

Newell, Allen. *Unified Theories of Cognition*. Harvard University Press, 1990.

Nicolescu, Monica N. and Matarić, Maja J. Natural Methods for Robot Task Learning: Instructive Demonstrations, Generalization and Practice. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 241–248, 2003.

O'Reilly, Randall C. and Munakata, Yuko. *Computational Explorations in Cognitive Neuroscience - Understanding the Mind by Simulating the Brain*. MIT Press, 2000.

Osoba, Osonde and Kosko, Bart. Noise-enhanced clustering and competitive learning algorithms. *Neural Networks*, 37:132–140, 2013.

Oudeyer, Pierre-Yves; Kaplan, Frédéric, and Hafner, Verena V. Intrinsic Motivation Systems for Autonomous Mental Development. *IEEE Transactions on Evolutionary Computation*, 11(2):265–286, 2007.

Papageorgiou, Constantine P.; Oren, Michael, and Poggio, Tomaso. A General Framework for Object Detection. In *Proceedings of the Sixth International Conference on Computer Vision*, pages 555–562, 1998.

Parkin, A. J. *Essential Cognitive Psychology*. Psychology Press, 2006.

Pearce, John M. and Bouton, Mark E. Theories Of Associative Learning In Animals. *Annual Review of Psychology*, 52:111–139, 2001.

Pfeifer, Rolf and Bongard, Josh. *How the Body Shapes the Way We Think: A New View of Intelligence*. MIT Press, 2006.

Pfeifer, Rolf and Scheier, Christian. *Understanding Intelligence*. MIT Press, 1999.

Porr, Bernd and Wörgötter, Florentin. Isotropic Sequence Order Learning. *Neural Computation*, 15(4):831–864, 2003.

Prudent, Yann and Ennaji, Abdellatif. An Incremental Growing Neural Gas Learns Topologies. In *Proceedings of the 2005 IEEE International Joint Conference on Neural Network*, volume 2, pages 1211–1216, 2005.

Ratcliff, Roger and McKoon, Gail. Memory Models. In Tulving, Endel and Craik, Fergus I. M., editors, *The Oxford Handbook of Memory*, chapter 35, pages 571–581. Oxford University Press, 2000.

Rauber, Andreas; Merkl, Dieter, and Dittenbach, Michael. The Growing Hierarchical Self-Organizing Map: Exploratory Analysis of High-Dimensional Data. *IEEE Transactions on Neural Networks*, 13(6):1331–1341, 2002.

Rescorla, Robert A. and Wagner, Allan R. A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. In Black, Abraham H. and Prokasy, William F., editors, *Classical Conditioning II: Current Theory and Research*, pages 64–99. Appleton-Century-Crofts, 1972.

Rizzuto, Daniel S. and Kahana, Michael J. Associative symmetry vs. independent associations. *Neurocomputing*, 32–33:973–978, 2000.

Rizzuto, Daniel S. and Kahana, Michael J. An Autoassociative Neural Network Model of Paired-Associate Learning. *Neural Computation*, 13(9):2075–2092, 2001.

Rosen, Virginia M. and Engle, Randall W. Forward And Backward Serial Recall. *Intelligence*, 25(1):37–47, 1997.

Rubin, David C. and Wenzel, Amy E. One hundred years of forgetting: A quantitative description of retention. *Psychological Review*, 103(4):734–760, 1996.

Rumelhart, David E.; Durbin, Richard; Golden, Richard, and Chauvin, Yves. Backpropagation: The Basic Theory. In Chauvin, Yves and Rumelhart, David E., editors, *Backpropagation: Theory, Architectures, and Applications*, Developments in Connectionist Theory, chapter 1. Psychology Press, 1995.

Saunders, Joe; Nehaniv, Chrystopher L., and Dautenhahn, Kerstin. Teaching Robots by Moulding Behavior and Scaffolding the Environment. In *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction*, pages 118–125, 2006.

Sharkey, Noel E. and Sharkey, Amanda J. C. An Analysis of Catastrophic Interference. *Connection Science*, 7(3 & 4):301–329, 1995.

Shen, Furao; Yu, Hui; Kasai, Wataru, and Hasegawa, Osamu. An Associative Memory System for Incremental Learning and Temporal Sequence. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1–8, 2010.

Shen, Furao; Ouyang, Qiubao; Kasai, Wataru, and Hasegawa, Osamu. A general associative memory based on self-organizing incremental neural network. *Neurocomputing*, 104:57–71, 2013.

Solway, Alec; Murdock, Bennet B., and Kahana, Michael J. Positional and temporal clustering in serial order memory. *Memory & Cognition*, 40(2):177–190, 2012.

Song, Sen; Miller, Kenneth D., and Abbott, L. F. Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*, 3(9): 919–926, 2000.

Strickert, Marc and Hammer, Barbara. Neural Gas for Sequences. In *Proceedings of the Workshop on Self-Organizing Networks*, pages 53–57, Kyushu Institute of Technology, 2003.

Sudo, Akihito; Sato, Akihiro, and Hasegawa, Osamu. Associative Memory for Online Learning in Noisy Environments Using Self-Organizing Incremental Neural Network. *IEEE Transactions on Neural Networks*, 20(6):964–972, 2009.

Sun, Ron. Desiderata for cognitive architectures. *Philosophical Psychology*, 17(3): 341–373, 2004.

Sutton, Richard S. and Barto, Andrew G. Toward a Modern Theory of Adaptive Networks: Expectation and Prediction. *Psychological Review*, 88(2):137–170, 1981.

Sutton, Richard S. and Barto, Andrew G. Time-Derivative Models of Pavlovian Reinforcement. In Gabriel, Michael R. and Moore, John W., editors, *Learning and Computational Neuroscience: Foundations of Adaptive Networks*, chapter 12. MIT Press, 1990.

Sutton, Richard S. and Barto, Andrew G. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

Taatgen, Niels and Anderson, John R. The Past, Present, and Future of Cognitive Architectures. *Topics in Cognitive Science*, 2(4):693–704, 2009.

Taddeo, Mariarosaria and Floridi, Luciano. Solving the symbol grounding problem: a critical review of fifteen years of research. *Journal of Experimental & Theoretical Artificial Intelligence*, 17(4):419–445, 2005.

Tan, Ah-Hwee; Carpenter, Gail A., and Grossberg, Stephen. Intelligence Through Interaction: Towards a Unified Theory for Learning. In Liu, Derong; Fei, Shumin; Hou, Zeng-Guang; Zhang, Huaguang, and Sun, Changyin, editors, *Advances in Neural Networks - ISNN 2007*, volume 4491 of *Lecture Notes in Computer Science*, pages 1094–1103. Springer Berlin Heidelberg, 2007.

Tangruamsub, Sirinart; Kawewong, Aram; Tsuboyama, Manabu, and Hasegawa, Osamu. Self-Organizing Incremental Associative Memory-Based Robot Navigation. *IEICE Transactions on Information and Systems*, E95-D(10):2415–2425, 2012.

Thórisson, Kristinn R. A New Constructivist AI: From Manual Methods to Self-Constructive Systems. In Wang, Pei and Goertzel, Ben, editors, *Theoritical*

Foundations of Artificial General Intelligence, chapter 9, pages 145–171. Atlantic Press, 2012.

Timmis, Jon and Neal, Mark. A resource limited artificial immune system for data analysis. *Knowledge-Based Systems*, 14(3–4):121–130, 2001.

Timmis, Jon; Neal, Mark, and Hunt, John. An artificial immune system for data analysis. *Biosystems*, 55(1–3):143–150, 2000.

van Rossum, M. C. W.; Bi, G. Q., and Turrigiano, G. G. Stable Hebbian Learning from Spike Timing-Dependent Plasticity. *The Journal of Neuroscience*, 20(23): 8812—8821, 2000.

Vargas, Patricia A.; Ho, Wan Ching; Lim, Mei Yii; Enz, Sibylle, and Aylett, Ruth. To Forget or Not to Forget: Towards a Roboethical Memory Control. In *Proceedings of the AISB 2009 Symposium on Killer Robots or Friendly Fridges: the Social Understanding of Artificial Intelligence*, 2009.

Vargas, Patricia A.; Fernaeus, Ylva; Lim, Mei Yii; Enz, Sibylle; Ho, Wan Chin; Jacobsson, Mattias, and Ayllet, Ruth. Advocating an ethical memory model for artificial companions from a human-centred perspective. *AI & Society*, 26(4): 329–337, 2011.

Vargas, Patricia A.; Di Paolo, Ezequiel A.; Harvey, Inman, and Husbands, Phil. *The Horizons of Evolutionary Robotics*. Intelligent Robotics & Autonomous Agents Series. MIT Press, 2014.

Varsta, Markus and Heikkonen, Jukka. Context Learning with the Self-Organizing Map. Technical report, Laboratory of Computational Engineering, Helsinki University of Technology, 1997.

Vavrečka, Michal and Farkaš, Igor. A Multimodal Connectionist Architecture for Unsupervised Grounding of Spatial Language. *Cognitive Computation*, 6:101–112, 2014.

Velik, Rosemarie. A Model for Multimodal Humanlike Perception based on Modular Hierarchical Symbolic Information Processing, Knowledge Integration, and Learning. In *Proceedings of the 2nd International Conference on Bio-Inspired Models of Network, Information, and Computing Systems*, pages 168–175, 2007.

Velik, Rosemarie and Bruckner, Dietmar. Neuro-Symbolic Networks: Introduction to a New Information Processing Principle. In *Proceedings of the 6th IEEE International Conference on Industrial Informatics*, pages 1042–1047, 2008.

Villmann, Thomas and Bauer, Hans-Ulrich. Applications of the growing self-organizing map. *Neurocomputing*, 21:91–100, 1998.

Viola, Paul and Jones, Michael. Rapid Object Detection using a Boosted Cascade of Simple Features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, 2001.

Wagner, Allan R. SOP: A Model of Automatic Memory Processing in Animal Behavior. In Spear, Norman E. and Miller, Ralph R., editors, *Information Processing in Animals: Memory Mechanisms*, chapter 1. Lawrence Erlbaum Associates, 1981.

Warmuth, Manfred K. and Kuzmin, Dima. Randomized Online PCA Algorithms with Regret Bounds that are Logarithmic in the Dimension. *Journal of Machine Learning Research*, 9:2287–2320, 2008.

Welch, Bernard Lewis. The Generalization of 'Student's' Problem when several different Population Variances are involved. *Biometrika*, 34(1–2):28–35, 1947.

Wichert, Andreas. Sub-Symbols and Icons. *Cognitive Computation*, 1:342–347, 2009.

Williams, Ronald J. and Zipser, David. Gradient-based Learning Algorithms for Recurrent Networks and Their Computational Complexity. In Chauvin, Yves and Rumelhart, David E., editors, *Backpropagation: Theory, Architectures, and Applications*, Developments in Connectionist Theory, chapter 13. Psychology Press, 1995.

Witten, Ian; Frank, Eibe, and Hall, Mark. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers, 3rd edition, 2011.

Wixted, John T. and Ebbesen, Ebbe B. On the form of forgetting. *Psychological Science*, 2(6):409–415, 1991.

Wixted, John T. and Ebbesen, Ebbe B. Genuine power curves in forgetting: A quantitative analysis of individual subject forgetting functions. *Memory & Cognition*, 25(5):731–739, 1997.

Wörgötter, Florentin and Porr, Bernd. Temporal Sequence Learning, Prediction, and Control – A Review of different models and their relation to biological mechanisms. *Neural Computation*, 17:245–319, 2005.

Yu, Hui; Shen, Furao, and Hasegawa, Osamu. A Multidirectional Associative Memory Based on Self-organizing Incremental Neural Network. In *Neural Information Processing. Models and Applications*, volume 6444 of *Lecture Notes in Computer Science*, pages 344–351. Springer Berlin Heidelberg, 2010.

Zhou, Junlin and Fu, Yan. Clustering High-Dimensional Data Using Growing SOM. In *Advances in Neural Networks – ISNN 2005*, volume 3497 of *Lecture Notes in Computer Science*, pages 63–68. Springer Berlin Heidelberg, 2005.